UNIT – I

COMPUTER SECURITY CONCEPTS

1.1. Introduction

The History of Information Security

- Began immediately after the first mainframes were developed
- Groups developing code-breaking computations during World War II created the first modern computers
- Physical controls to limit access to sensitive military locations to authorized personnel
- Rudimentary in defending against physical theft, espionage, and sabotage

The 1960s

- Advanced Research Procurement Agency (ARPA) began to examine feasibility of Redundant Networked Communications
- Larry Roberts developed ARPANET from its inception

The 1970s and 80s

- ARPANET grew in popularity as did its potential for misuse
- Fundamental problems with ARPANET security were identified
  - No safety procedures for dial-up connections to ARPANET
  - Non-existent user identification and authorization to system
- Late 1970s: microprocessor expanded computing capabilities and security threats

R-609

- Information security began with Rand Report R-609 (paper that started the study of computer security)
- Scope of computer security grew from physical security to include:
  - Safety of data
  - Limiting unauthorized access to data
  - Involvement of personnel from multiple levels of an organization

The 1990s

- Networks of computers became more common; so too did the need to interconnect networks
- Internet became first manifestation of a global network of networks
- In early Internet deployments, security was treated as a low priority

The Present

- The Internet brings millions of computer networks into communication with each other—many of them unsecured
  - Ability to secure a computer's data influenced by the security of every computer to which it is connected

1.2. The need for security

- **"**The quality or state of being secure—to be free from danger"
- A successful organization should have multiple layers of security in place:

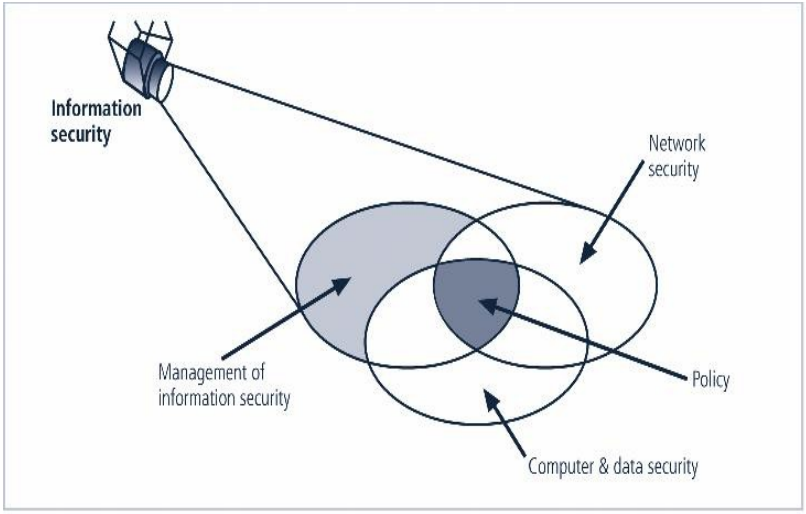
– Physical security
– Personal security
– Operations security
– Communications security
– Network security
– Information security

1.3. Security approaches and Security Principles

What is Information Security?

The protection of information and its critical elements, including systems and hardware that use, store, and transmit that information.

- Necessary tools: policy, awareness, training, education, technology.
- C.I.A. triangle was standard based on confidentiality, integrity, and availability.
- C.I.A. triangle now expanded into list of critical characteristics of information.



Components of Information Security

Information Security: It can be defined as "measures adopted to prevent the unauthorized use, misuse, modification or denial of use of knowledge, facts, data or capabilities". Three aspects of IS are:

Security Attack: Any action that comprises the security of information

Security Mechanism: A mechanism that is designed to detect, prevent, or recover from a security.

Security Service:

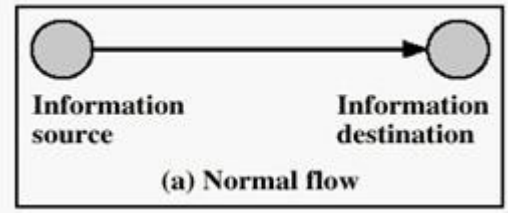
It is a processing or communication service that enhances the security of the data processing systems and information transfer. The services are intended to counter security attacks by making use of one or more security mechanisms to provide the service.

1.4. **Security Attacks:**

We can classify the types of attacks into four types such as

- Interruption
- Interception
- Modification
- Fabrication

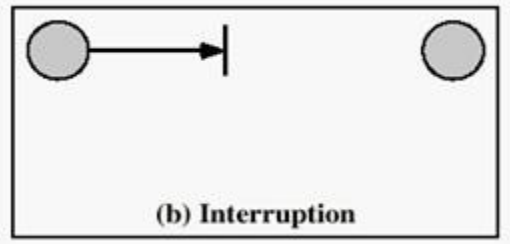
The normal flow of information transmission will be like this.



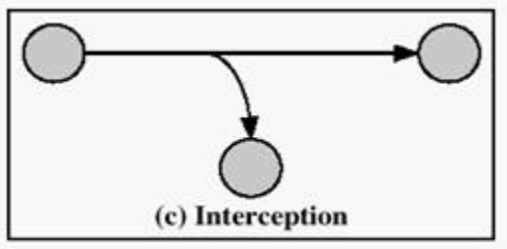
Interruption: This is also known as attack on availability. This is an attack where the sender data is somehow interrupted by the intruder so that it does not reach the destination.

For example, a collection of cooperative intruders would bombard the destination with bulk of messages so that the destination will not get chance to process the sender message.

The following figure illustrate this.

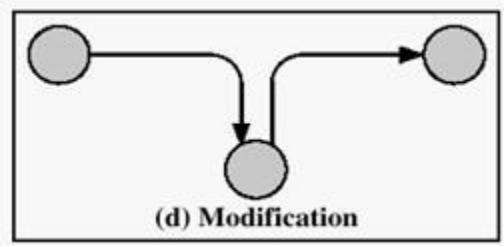


Interception: This is also known as attack on confidentiality. It is illustrated below.



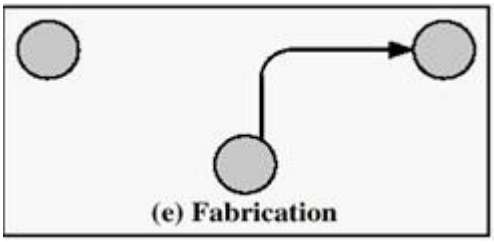
In this kind of attack , intruder would read the message of sender. Here, intruder does not do any harm for the data/message. Detecting this kind of intruders is difficult.

Modification: This is also known as an attack on integrity. This is illustrated below.



In this kind of attack , intruder read and modify the sender data/message and then send the modified message to the receiver.

Fabrication: This is also known as an attack on authenticity



This is the kind of attack where intruder will send the data to the receiver as if he is original sender.
These attacks are further classified into two types such as passive attacks and active attacks.

a. Passive attacks:

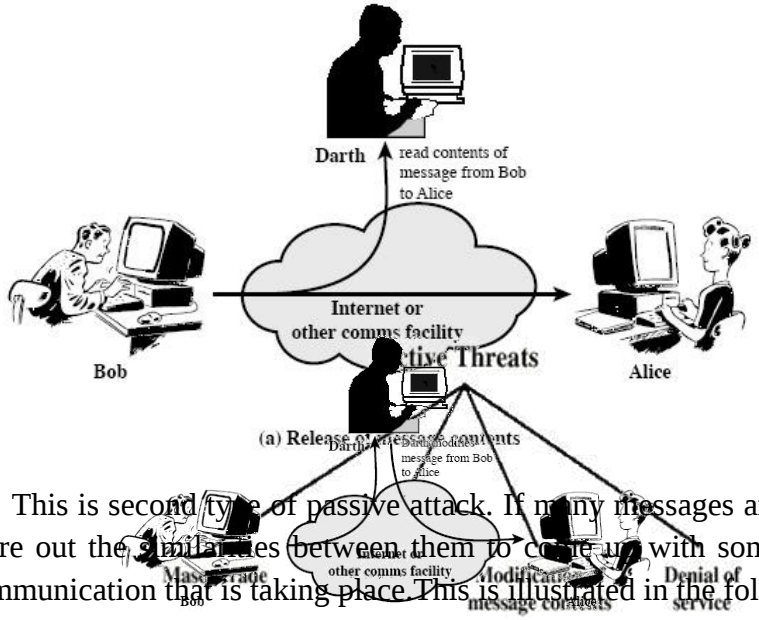
Passive attacks are those wherein the attacker monitor the data transmission. So, the attacker aims to obtain the information that is being transmitted. The term passive indicates that the attacker does not attempt to modify the data.Passive attacks are further classified into two types such as

- Release of message contents
- Traffic analysis

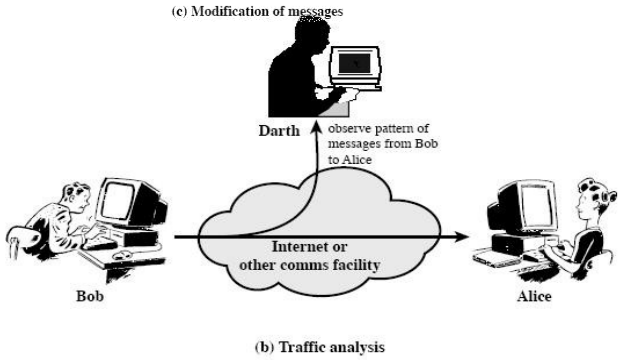
It is illustrated below.

Release of message contents: It is quite simple to understand. When we send a confidential email to our friend, we desire that only she be able to access it. Otherwise, the contents of the message are released to someone else.

Using certain security mechanism , we can prevent this. This illustrated in the following figure.



Traffic analysis: This is second type of passive attack. If many messages are passing through, a passive attacker could try to figure out the similarities between them to come up with some sort of pattern that provides clues regarding the communication that is taking place.This is illustrated in the following figure.

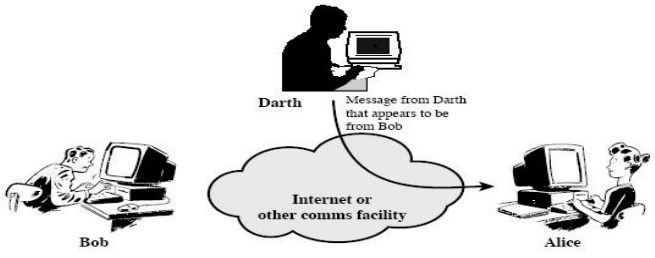


(b) Traffic analysis

b. Active attacks:

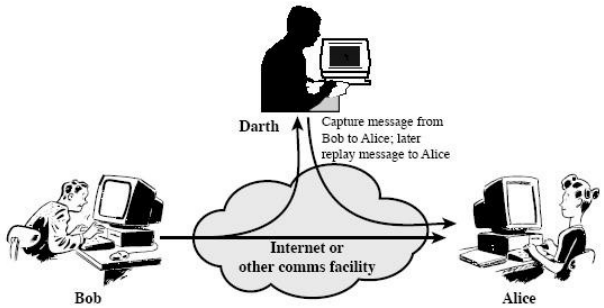
The active attacks involve some modification of the data stream. These attacks can not be prevented easily. These attacks are further classified into four types . It is illustrated in the following figure.

Masquerade attack: It takes place when one entity pretend to be a different entity. It is illustrated in the following figure. It resembles attack on authentication.



(a) Masquerade

Replay attack: It involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect. It is illustrated in the following figure.



(b) Replay

Modification of message simply means that some portion of the legitimate message is altered, delayed or reordered. It is illustrated below.

The denial of service prevents the normal use or management of communication facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination..It is illustrated below.

(d) Denial of service

1.5. Security Services:

X.800 standard define the following security services. These services are categorized into five types such as

1. Confidentiality (privacy)
2. Authentication (who created or sent the data)
3. Integrity (has not been altered)
4. Non-repudiation (the order is final)
5. Access control (prevent misuse of resources)
6. Availability (permanence, non-erasure)

– Denial of Service Attacks
– Virus that deletes files

1. **Confidentiality**: Confidentiality is nothing but the protection of transmitted data from passive attacks. With respect the content several levels of protection can be identified.

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.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that the attacker should be able to see the traffic on the specified path.

2. **Authentication:** The authentication service is concerned with giving assurance that the communication is authentic.

Authentication ensures that the receiver is receiving the message/data from intended party only. It also ensures that the message/data is not modified by any intruder.

3. **Access control** :

Access control is the ability to limit and control the access to host systems and applications.

4. **Data Integrity**:

Assurance that data received is as sent by an authorized entity

It allows the recipient of a message to verify it has not been modified in transit.

5. **Non-repudiation**:

Protection against denial by one of the parties in a communication

Makes it difficult for the originator of a message to falsely deny later that they were the party that sent the message.

6. **Availability**

It ensures that a service or information is available to an (authorized) user upon demand and without delay. *Denial of Service (DoS) attacks* seek to interrupt a service or make some information unavailable to legitimate users.

1.6. Security mechanisms:

No single mechanism can provide all the security services wanted. But encryption or encryption-like information transformation is a key enabling technology.

Security Mechanisms: We have various security mechanisms to achieve information security:

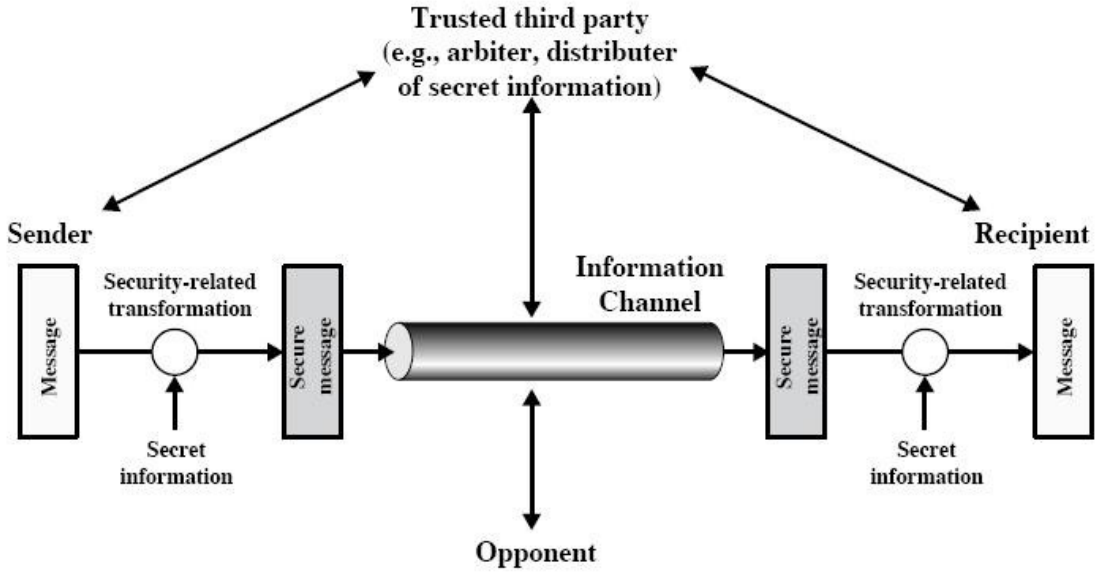
1. **Access Control** : Access control mechanism will control the data access thru access rights. It uses the mechanisms such as Discretionary Access control mechanism and Mandatory Access control mechanism.
2. **Encipherment:** It uses mathematical algorithms to transform data into a unreadable format. The transformation and subsequent recovery of the data depend on an algorithms and zero or more encryption keys.
3. **Digital signatures:** Digital signature is an electronic signature created thru data to be transferred. Digital signature will allow the user to check for the forgery.
4. **Data integrity**: Data integrity ensures that the message is received as it is sent.
5. **Authentication exchange**: A mechanism used to ensure the identity of sender
6. **Traffic padding**: The insertion of bits into gaps in a data steam to frustrate traffic analysis attempts.
7. **Routing control:** Enables selection of safest route.

Relationship between Security Services and Mechanisms

Service	Mechanism							
	Enciph-erment	Digital signature	Access control	Data integrity	Authenti-cation exchange	Traffic padding	Routing control	Notari-zation
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Non-repudiation		Y		Y				Y
Availability				Y	Y			

A Model for Network Security:

A model for network security is shown in the following figure:



A message is to be transferred from one party to another across some sort of network.
All the techniques for providing security have the following components:

A security related transformation on the information to be sent Examples include the encryption of the message.

Some secret information shared by the two principals and it is hope , unknown to the opponent. For example, an encryption key is used in conjunction with the transformation to encrypt the message before transmission and decrypt the message at the recipient.

A trusted third party may be needed to achieve secure transmission.

2. Cryptography Concepts and Techniques

2.1. Introduction

Cryptography
A cipher is a secret method of writing as by code.

Cryptography is the study of techniques related to aspects of information security. Hence cryptography is concerned with the writing (ciphering or encoding) and deciphering (decoding) of messages in secret code.

2.2. Plain text and Ciphertext

The different terms we find in Cryptography are

Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.
Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.

Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.

The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

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

Cryptographic systems are classified along three independent dimensions:

1. The type of operations used for performing plaintext to ciphertext

All the encryption algorithms make use of two general principles; substitution and transposition through which

plaintext elements are rearranged. Important thing is that no information should be lost.

2. The number of keys used

If single key is used by both sender and receiver, it is called symmetric, single-key, secret-key or conventional encryption. If sender and receiver each use a different key, then it is called asymmetric, two-key or public-key encryption.

3. The way in which plaintext is processed

A block cipher process the input as blocks of elements and generated an output block for each input block. Stream cipher processes the input elements continuously, producing output one element at a time as it goes along.

Substitution: Method by which units of plaintext are replaced with ciphertext according to a regular system.
Transposition: Here, units of plaintext are rearranged in a different and usually quite complex order, but the unit themselves are left unchanged.

Cryptanalysis

The process of attempting to discover the plaintext or key is known as cryptanalysis. It is very difficult when only the ciphertext is available to the attacker as in some cases even the encryption algorithm is not known. The most common attack under these circumstances is brute-force approach of trying all the possible keys. This attack is made impractical when the key size is considerably large. The table below gives an idea on types of attacks on encrypted messages.

Type of Attack	Known to Cryptanalyst
Ciphertext only	Encryption algorithm
	Ciphertext to be decoded
Known plaintext	Encryption algorithm
	Ciphertext to be decoded
	One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	Encryption algorithm
	Ciphertext to be decoded
	Plaintext message chosen by cryptanalyst, together with its ciphertext generated with the secret key
Chosen ciphertext	Encryption algorithm
	Ciphertext to be decoded
	Purported (supposed) ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Chosen text	Encryption algorithm
	Ciphertext to be decoded
	Plaintext message chosen by cryptanalyst, together wit h its ciphertext generated with the secret key
	Purported (supposed) ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

- The most difficult problem is when all that is available is the ***ciphertext*** *only*.

  - In some cases, not even the encryption algorithm is known, but in general we can assume that the opponent traces 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 ciphertext itself, generally applying various statistical tests to it. To use this approach, the opponent must have some general idea of the type of plaintext that is concealed (masked or hidden), such as English or French text, an EXE file, a

    Java source listing, an accounting file, and so on.

The ciphertext only attack is the easiest to defend against because the opponent has the least am ount of information to work with.

In many cases, however, the analyst has more information. The analyst may be able to capture one or more plaintext messages as well as their encryptions. Or the analyst may know that certain plaintext patterns will appear in a message.

For example, a file that is encoded in the Postscript format always begins with the same pattern, or there may be a standardized header or banner to an electronic funds transfer message, and so on.

All these are examples of ***known plaintext***. With this knowledge, the analyst may be able to deduce (work out) the key on the basis of the way in which the known plaintext is transformed.

Closely related to the ***known plaintext*** attack is what might be referred to as a probable word attack. If the opponent is working with the encryption of some general prose (text) message, he or she may have little

knowledge of what is in the message. However, if the opponent is after some very specific information, then parts of the message may be known.

For example, if an entire accounting file is being transmitted, the opponent may know the placement of certain key words in the header of the file. As another example, the source code for a program developed by a corporation might include a copyright statement in some standardized position.

If the analyst is somehow able to get the source system to insert into the sys tem a message chosen by the analyst, then a chosen plaintext attack is possible.

In general, if the analyst is able to choose the messages to encrypt, th e analyst may deliberately (carefully) pick patterns that can be expected to reveal the structure of the key.

The other two other types of attacks are: ***chosen ciphertext and chosen text***. These are less commonly employed as cryptanalytic techniques but are nevertheless (however) possible avenues (ways) of attack. Only relatively weak algorithms fail to withstand a ciphertext -only attack. Generally, an encryption algorithm is designed to withstand a known-plaintext attack.

An encryption scheme is **computationally secure** if the ciphertext generated by the scheme meets one or both of the following criteria:

- The cost of breaking the cipher exceeds the value of the encrypted information.
- The time required to break the cipher exceeds the useful lifetime of the info rmation. Unfortunately it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully.

However, assuming there are **no inherent** mathematical weaknesses in the algorithm, then a brute -force approach is indicated, and here we can make some reasonable estimates about costs and time.

A brute-force approach involves trying every possible key until an intelligible (clear) translation of the ciphertext into plaintext is obtained. On average, half of all pos sible keys must be tried to achieve success.

Table 2.2 shows how much time is involved for various key sizes.

The 56-bit key size is used with the DES

Key size (bits)	Number of alternative keys		Time required at 1 decryption/μs		Time required at 10^6 decryption/μs
32	2^{32}	$= 4.3 \times 10^9$	$2^{31}\ \mu s$	= 35.8 minutes	2.15 milliseconds
56	2^{56}	$= 7.2 \times 10^{16}$	$2^{55}\ \mu s$	= 1142 years	10.01 hours
128	2^{128}	$= 3.4 \times 10^{38}$	$2^{127}\ \mu s$	$= 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	2^{168}	$= 3.7 \times 10^{50}$	$2^{167}\ \mu s$	$= 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	26!	$= 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s$	$= 6.4 \times 10^{12}$ years	6.4×10^6 years

Table: Average time required for Exhaustive Key Search

For each key size, the results are shown assuming that it takes 1µs to perform a single decryption, which is a reasonable order of magnitude for today's machines.

The final column of Table 2.2 considers the results for a system that can process 1 million keys per microsecond. As you can see, at this performance level, DES can no longer be considered computationally secure.

2.3. Substitution:

Method by which units of plaintext are replaced with ciphertext according to a regular system.

These techniques involve substituting or replacing the contents of the plaintext by other letters, numbers or symbols. Different kinds of ciphers are used in substitution technique.

Caesar Ciphers:

It is the oldest of all the substitution ciphers. A Caesar cipher replaces each letter of the plaintext with an alphabet. Two examples can be given: 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 Choose k, Shift all letters by k
For example, if k = 5
A becomes F, B becomes G, C becomes H, and so on…

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

$c = E(p) = (p + k) \bmod (26)$ $p = D(c) = (c - k) \bmod (26)$ With a Caesar cipher, there are only 26 possible keys, of which only 25 are of any use, since mapping A to A etc doesn't really obscure the message

Mono alphabetic Ciphers:

Here, Plaintext characters are substituted by a different alphabet stream of characters shifted to the right or left by n positions. When compared to the Caesar ciphers, these mono alphabetic ciphers are more secure as each letter of the ciphertext can be any permutation of the 26 alphabetic characters leading to 26! or greater than 4 x 1026 possible keys. But it is still vulnerable to cryptanalysis, when a cryptanalyst is aware of the nature of the plaintext, he can find the regularities of the language. To overcome these attacks, multiple substitutions for a single letter are used. For example, a letter can be substituted by different numerical cipher symbols such as 17, 54, 69….. etc. Even this method is not completely secure as each letter in the plain text affects on letter in the ciphertext. Or, using a common key which substitutes every letter of the plain text. The key *ABCDEFGHIIJ KLMNOPQRSTUVWXYZ QWERTYUIIOPAS DFGHJ KLZXCV BNM* Would encrypt the message *II think therefore II am* into ***OZIIOFAZIITKTYGKTOQD***

But any attacker would simply break the cipher by using frequency analysis by observing the number of times each letter occurs in the cipher text and then looking upon the English letter frequency table. So, substitution cipher is completely ruined by these attacks. Monoalphabetic ciphers are easy to break as they reflect the frequency of the original alphabet. A countermeasure is to provide substitutes, known as homophones for a single letter.

Playfair Ciphers:

It is the best known multiple –letter encryption cipher which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair Cipher is a digram substitution cipher offering a relatively weak method of encryption. It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians and Germans during World War II. This was because Playfair is reasonably fast to use and requires no special equipment. A typical scenario for Playfair use would be to protect important but non-critical secrets during actual combat. By the time the enemy cryptanalysts could break the message, the information was useless to them. It is based around a 5x5 matrix, a copy of which is held by both communicating parties, into which 25 of the 26 letters of the alphabet (normally either j and i are represented by the same letter or x is ignored) are placed in a random fashion. For example, the plain text is *Shi Sherry loves Heath Ledger* and the agreed key is *sherry*. The matrix will be built according to the following rules. in pairs, without punctuation,

All Js are replaced with Is.

SH IS HE RR YL OV ES HE AT HL ED GE R
Double letters which occur in a pair must be divided by an X or a Z.

SH IS HE RX RY LO VE SH EA TH LE DG ER The alphabet square is prepared using, a 5*5 matrix, no repetition

letters, no Js and key is written first followed by the remaining alphabets with no i and j.

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

For the generation of cipher text, there are three rules to be followed by each pair of letters.

Letters appear on the same row replace them with the letters to their immediate right respectively
Letters appear on the same column replace them with the letters immediately below respectively
Not on the same row or column replace them with the letters on the same row respectively but at the other pair

of corners of the rectangle defined by the original pair.
Based on ***HE GH ER DR YS IQ WH***

HE SC OY KR AL RY

Another example which is simpler than the above one can be given as:
Here, key word is *playfair*. Plaintext is *Hellothere hellothere* becomes-----*he lx lo th er ex* .

Applying the rules again, for each pair, If they are in the same row, replace each with the letter to its right (mod 5) *he KG* If they are in the same column, replace each with the letter below it (mod 5) *lo RV* Otherwise, replace each with letter we'd get if we swapped their column indices *lx YV*

So the cipher text for the given plain text is **KG YV RV QM GI KU** To decrypt the message, just reverse the process. Shift up and left instead of down and right. Drop extra x's and locate any missing I's that should be j's.

The message will be back into the original readable form. no longer used by military forces because of the advent of digital encryption devices.

Playfair is now regarded as insecure for any purpose because modern hand-held computers could easily break the cipher within seconds.

2.4. Transposition:

Here, units of plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged.

A **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are

commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed. Transposition ciphers encrypt plaintext by moving small pieces of the message around. Anagrams are a primitive transposition cipher. This table shows "VOYAGER" being encrypted with a primitive transposition cipher where every two letters are switched with each other:

V	O	Y	A	G	E	R
O	V	A	Y	E	G	R

Another simple example for transposition 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, write the message “meet me after the toga party” out as:

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

MEMATRHTGPRYETEFETEOAA
T

The following example shows how a pure permutation cipher could work: You write your plaintext message along the rows of a matrix of some size. You generate ciphertext by reading along the columns. The order in which you read the columns is determined by the encryption key:

Ciphertext: ***TITESMAIRDEMHHEENOOYETGTI*** The cipher can be made more secure by performing multiple rounds of such permutations.

```
key:            2 5 3 1 6 4

plaintext:      m e e t m e
                a t m i d n
                i g h t f o
                r t h e g o
                d i e s x y
```

2.5. Symmetric Key Cryptography

Symmetric encryption, also referred to as conventional encryption, secret -key, or single-key encryption, was the only type of encryption in use prior to the development of public-key encryption in the late 1970s. It remains by far the most widely used of the two types of encryption.

In this unit we begin with the general model for the symmetric encryp tion process: this enables us to understand

the context within which the algorithms are used.

Then we look at three important encryption algorithms: DES, triple DES, and AES. We then examine the application of these algorithms to achieve confidentiality.

CONVENTIONAL ENCRYPTION PRINCIPLES

A symmetric encryption scheme has *five* ingredients (components)

- **Plain text:** This is the original 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 algorithm.
  - The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output.
  - It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This is the encryption algorithm run in reverse.
  - It takes the ciphertext and the same secret key and produces the original plaintext.

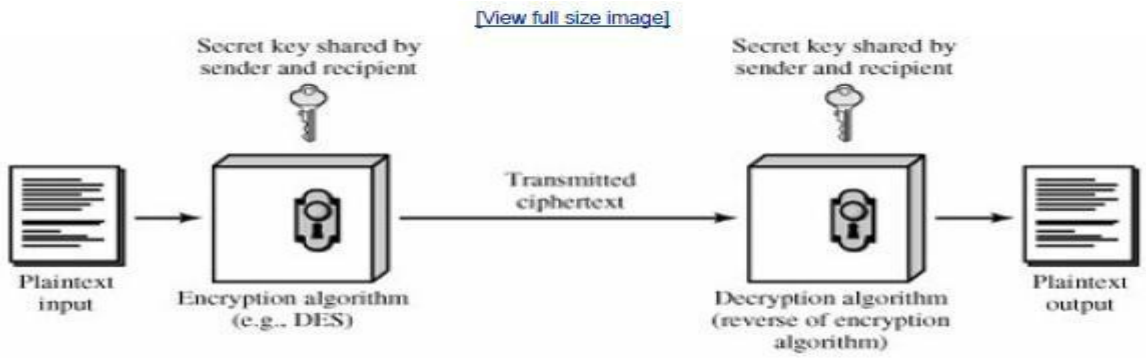


Fig 2.1: Simplified Model of Conventional Encryption

There are two requirements for secure use of symmetric encryption;

1. **W e need a strong encryption algorithm**. The algorithm should be in such a way that an opponent who knows the algorithm and has access to one or more ciphertexts will not be able 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 (control) 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 usi ng this key is readable.

The security of symmetric encryption depends on the secrecy of the key, not the secrecy of the algorithm. That is, it is not possible 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 makes it feasible (possible) for widespread use.

But with the use of symmetric encryption, t h e pr i n ci p al security problem is maintaining the secrecy of the key.

2.6. Asymmetric 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:

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

It is **asymmetric** because those who encrypt messages or verify signatures cannot decrypt messages or create

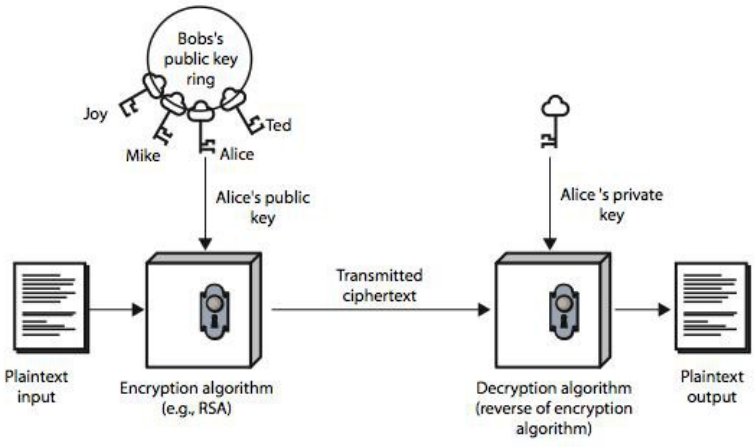
signatures

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

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

Either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms like RSA)

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



(a) Encryption

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.

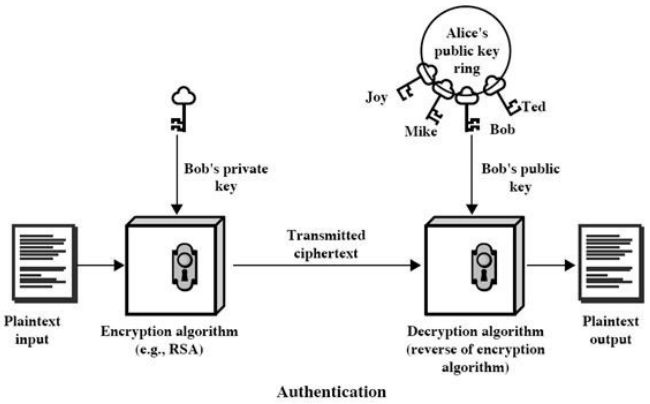
If B wishes to send a plain message X to A, then he sends the ciphertext Y=E(KUA,X) The intended receiver A will decrypt the message: D(KRA,Y)=X

The first attack on Public-key Cryptography is the attack on Authenticity.

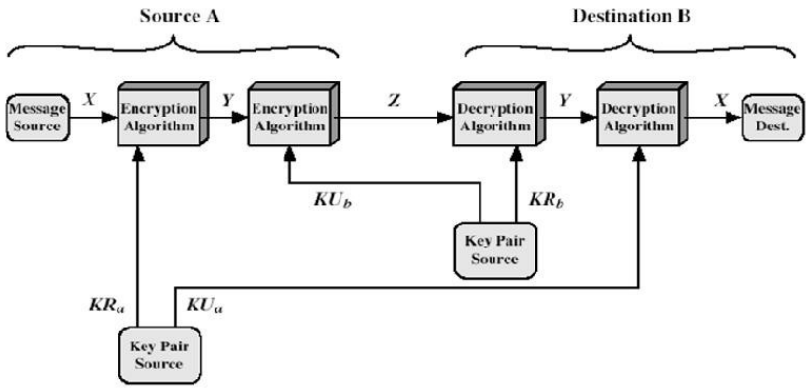
An attacker may impersonate user B: he sends a message E(KUA,X) and claims in the message to be B –A has no guarantee this is so. To overcome this, B will encrypt the message using his private key: Y=E(KRB,X).

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:

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:



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:



Public-Key Cryptosystem: Secrecy and Authentication

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 KUb, private key KRb).
2. Easy for sender A to generate ciphertext:
3. Easy for the receiver B to decrypt ciphertect 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 ciphertext C
6. Either of the two keys can be used for encryption, with the other used for decryption:

Easy is defined to mean a problem that can be solved in 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 S-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*** 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:*

Like private key schemes brute force **exhaustive search** attack is always theoretically possible. But keys used are too large (>512bits).

Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems. More generally the **hard** problem is known, its just made too hard to do in practise.

Requires the use of **very large numbers,** hence is **slow** compared to private key schemes

2.7. Steganography

Steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper's suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy-to-use communication channels for steganography. Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium's redundant bits (those that can be modified without destroying that medium's integrity). The embedding process creates a stego medium by replacing these redundant bits with data from the hidden message. Modern steganography's goal is to keep its mere presence undetectable, but steganographic systems— because of their invasive nature—leave behind detectable traces in the cover medium. Even if secret content is not revealed, the existence of it is: modifying the cover medium changes its statistical properties, so eavesdroppers can detect the distortions in the resulting stego medium's statistical properties. The process of finding these distortions is called statistical steganalysis.

*** * * * * * * *END OF UNIT I* * * * * * * ***

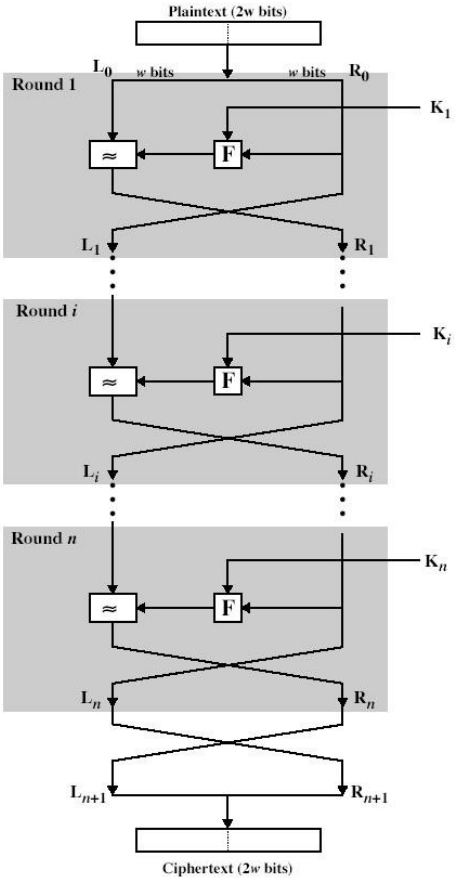
UNIT-II

Symmetric key ciphers:

2.1. Block cipher principles

Feistel Cipher Structure

Most symmetric block ciphers are based on a *Feistel Cipher Structure*. It was first described by Horst Feistel of IBM in 1973 and is still forms the basis for almost all conventional encryption schemes. It makes use of two properties namely *diffusion* and *confusion*; identified by Claude Shannon for frustrating statistical cryptanalysis. Confusion is basically defined as the concealment of the relation between the secret key and the cipher text. On the other hand, diffusion is regarded as the complexity of the relationship between the plain text and the cipher text.



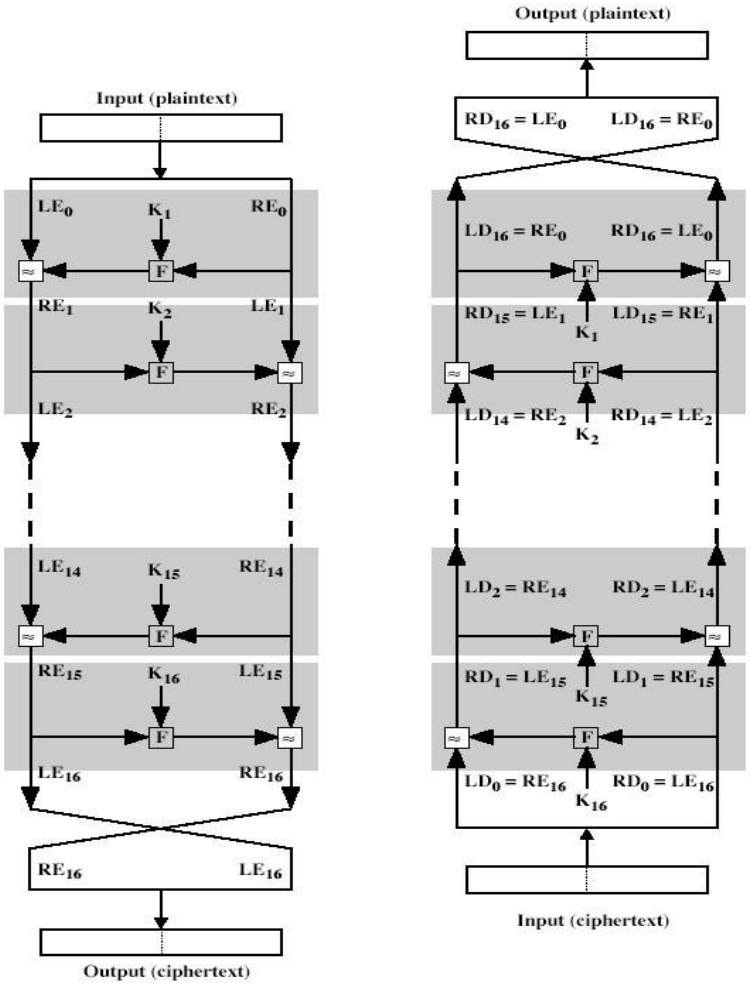
The function of Feistel Cipher is shown in the above figure and can be explained by following steps:
The input to the encryption algorithm is a plaintext block of length 2w bits and a key K.

The two halves pass through n rounds of processing and then combine to produce the cipher text block

Each Round i has inputs Li-1 and Ri-1, derived from the previous round, as well as a unique subkey Ki generated

by a sub-key generation algorithm.
Applying a round function F to right half of data and then taking XOR of the output of that function and left half of data. The round function F is common to every round but parameterized by round subkey Ki.
The structure is a particular form of substitution-permutation network (SPN) proposed by Shannon. The realization or development of a Feistel encryption scheme depends on the choice of the following parameters and design features:

• **Block size:** larger block sizes mean greater security but slower processing. Block size of 64 bits has been nearly universal in block cipher design.
• **Key Size:** larger key size means greater security but slower processing. Most common key length in modern algorithms is 128 bits.
• **Number of rounds:** multiple rounds offer increasing security but slows cipher. Typical size is 16 rounds.
• **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
• **Round Function:** greater complexity will make cryptanalysis harder.
• **Fast software en/decryption & ease of analysis:** are more recent concerns for practical use and testing.

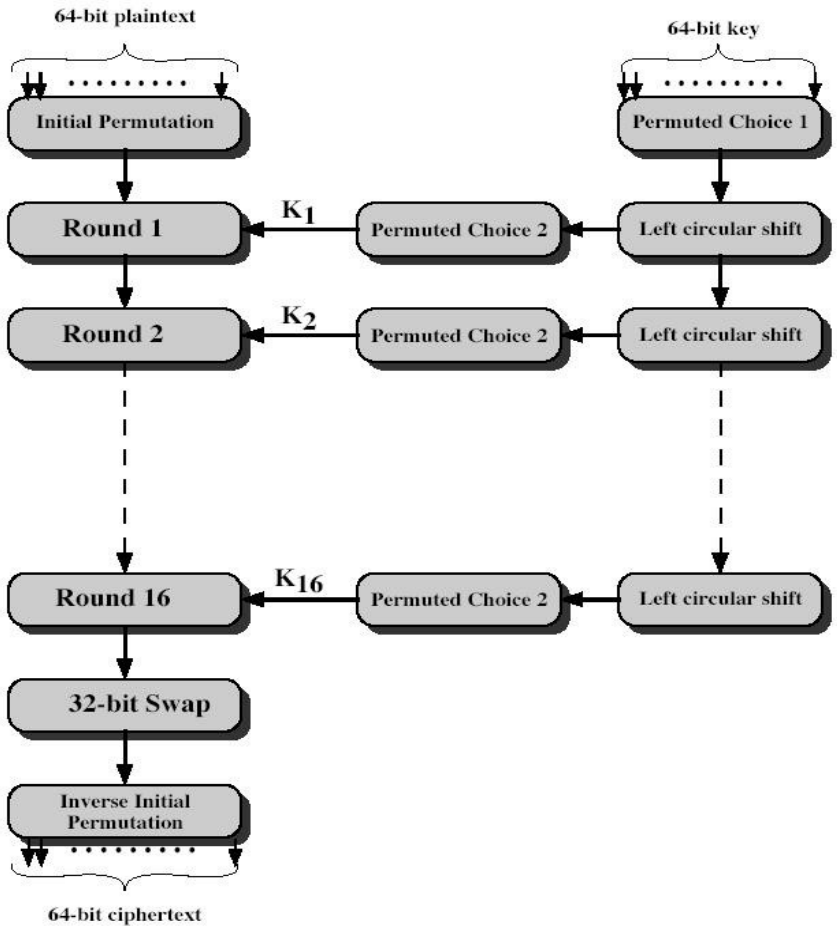


Feistel Cipher Decryption

The process of decryption with a Fiestel cipher is same as the encyption process. Use the ciphertext as input to the algorithm, but use the subkeys Ki in the reverse order. Use Kn in the first round and Kn-1 in the second round and so on until k1 is used in the last round. Main advantage is we need not implement two different algorithms for encryption and decryption. The Fiestel cipher has the advantage that encryption and decryption operations are very similar, even identical in some cases requiring only a reversal in the key schedule. Therefore, the size of the code or circuitry required to implement such a cipher is nearly halved.

2.2. Data Encryption Standard

In 1974, IBM proposed "Lucifer", an encryption algorithm using 64-bit keys. Two years later (1977), NBS (now NIST) in consultation with NSA made a modified version of that algorithm into a standard. DES uses the two basic techniques of cryptography - confusion and diffusion. At the simplest level, diffusion is achieved through numerous permutations and confusion is achieved through the XOR operation and the S-Boxes. This is also called an S-P network The DES encryption scheme can be explained by the following figure



The plain text is 64 bits in length and the key in 56 bits in length. Longer plain text amounts are processed in 64-bit blocks. The main phases in the left hand side of the above figure i.e. processing of the plain text are,

Initial Permutation (IP): The plaintext block undergoes an initial permutation. 64 bits of the block are

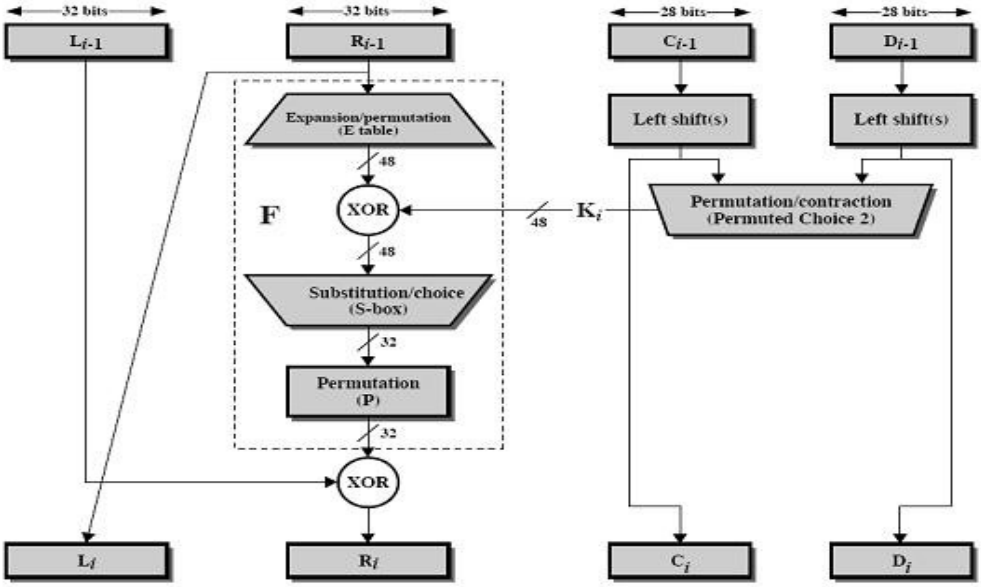
permuted.

A Complex Transformation: 64 bit permuted block undergoes 16 rounds of complex transformation. Subkeys are used in each of the 16 iterations.

32-bit swap: The output of 16th round consists of 64bits that are a function of input plain text and key.32 bit left and right halves of this output is swapped.

Inverse Initial Permutation (IP-1): The 64 bit output undergoes a permutation that is inverse of the initial permutation.
On the right hand side part of the figure, the usage of the 56 bit key is shown. Initially the key is passed through a permutation function. Now for each of the 16 iterations, a new subkey (Ki) is produced by combination of a left circular shift and a permutation function which is same for each iteration. A different subkey is produced because of repeated shifting of the key bits.

The following figure shows a closer view of algorithms for a single iteration. The 64bit permuted input passes through 16 iterations, producing an intermediate 64-bit value at the conclusion of each iteration.

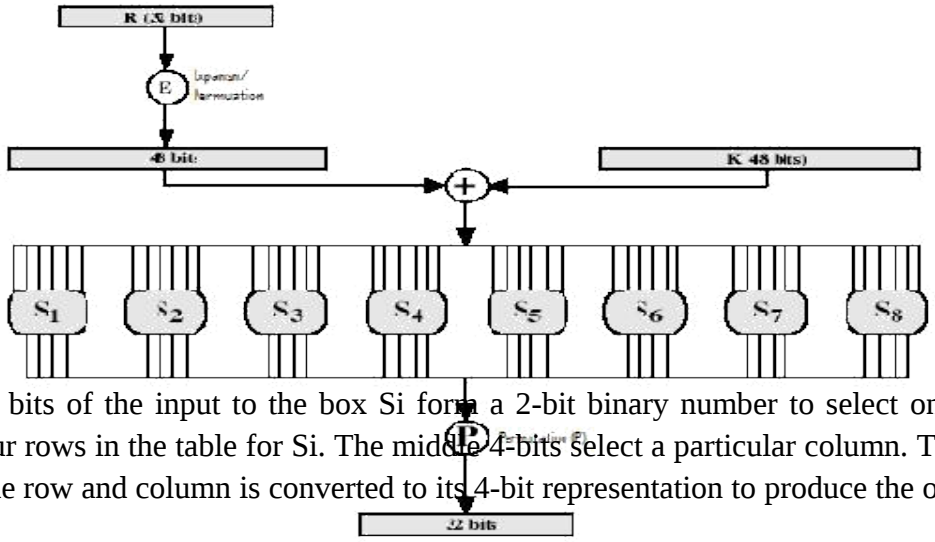


Single Round of DES Algorithm

The left and right halves of each 64 bit intermediate value are treated as separated 32-bit quantities labeled L (left) and R (Right). The overall processing at each iteration is given by following steps, which form one round in an S-P network.

The left hand output of an iteration (Li) is equal to the right hand input to that iteration Ri-1. The right hand output Ri is exclusive OR of Li-1 and a complex function F of Ri-1 and Ki. The function F can be depicted by the following figure. S1, S2-----S8 represent the ”S-boxes” , which maps each combination of 48 input bits into a

particular 32 bit pattern. For the generation of sub key of length 48 bits, a 56bit key is used which is first passed through a permutation function and then halved to get two 28 bit quantities labeled C0 and D0. At each iteration, these two C and D are subjected to a circular left shift or rotation of 1 or 2 bits. These shifted values serve as input to the next iteration and also to another permutation function which produces a 48-bit output. This output is fed as input to function **F(R i-1, Ki).**



The first and last bits of the input to the box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si. The middle 4-bits select a particular column. The decimal value in the cell selected by the row and column is converted to its 4-bit representation to produce the output.

The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. The process of decryption with DES is essentially the same as the encryption process: no different algorithm is used. The ciphertext is used as input to the DES algorithm and the keys are used in the reverse order i.e. K16 in the first iteration, K15 on the second iteration and so on until k1 is used on the sixteenth and last iteration.

Strength of DES: *Avalanche Effect*: An effect in DES and other secret key ciphers where each small change in plaintext implies that somewhere around half the ciphertext changes. The avalanche effect makes it harder to successfully cryptanalyze the ciphertext. DES exhibits a strong Avalanche effect. Concern about the strength of DES falls into two categories i.e. strength of algorithm itself and use of 56- bit key. Though many attempts were made over the years to find and exploit weaknesses in the algorithm, none of them were successful in discovering any fatal weakness in DES. A serious concern is with the key size as the time passed the security in DES became

getting compromised by the advent of supercomputers which succeeded in breaking the DES quickly using a brute-force attack. If the only form of attack that could be made on an encryption algorithm is brute force, the way of countering it is obviously using long keys. If a key of size 128 bits is used, it takes approximately 1018 years to break the code making the algorithm unbreakable by brute-force approach. The two analytical attacks on DES are Differential cryptanalysis and Linear cryptanalysis. Both make use of Known plaintext-ciphertext pairs and try to attack the round structure and the S-Boxes. Recent advancements showed that using Differential cryptanalysis, DES can be broken using 247 plaintext-ciphertext pairs and for linear cryptanalysis, the number is even reduced to 241.

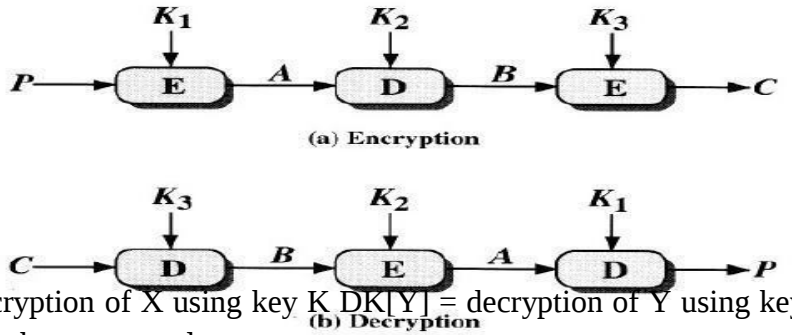
Triple DES

The first answer to problems of DES is an algorithm called Double DES which includes double encryption with two keys. It increases the key size to 112 bits, which seems to be secure. But, there are some problems associated with this approach. issue of reduction to single stage: In other words, could there be a key K3 such that EK2 (EK21(P))= EK3(P)? “meet-in-the-middle” attack:

X = EK1(P) = DK2(C)

Test the two keys for the second pair of plaintext-ciphertext and if they match, correct keys are found

Triple DES was the answer to many of the shortcomings of DES. Since it is based on the DES algorithm, it is very easy to modify existing software to use Triple DES. 3DES was developed in 1999 by IBM – by a team led by Walter Tuchman. 3DES prevents a meet-in-the-middle attack. 3DES has a 168-bit key and enciphers blocks of 64 bits. It also has the advantage of proven reliability and a longer key length that eliminates many of the shortcut attacks that can be used to reduce the amount of time it takes to break DES. 3DES uses three keys and three executions of the DES algorithm. The function follows an encrypt-decrypt-encrypt (EDE) sequence.



plaintext and EK[X] = encryption of X using key K DK[Y] = decryption of Y using key K Decryption is simply the same operation with the keys reversed

Triple DES runs three times slower than standard DES, but is much more secure if used properly. With three distinct keys, TDEA has an effective key length of 168 bits making it a formidable algorithm. As the underlying algorithm is DEA, it offers the same resistance to cryptanalysis as is DEA. Triple DES can be done using 2 keys

or 3 keys.

3.3. AES

The AES Cipher The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. The number of rounds is dependent on the key size i.e. for key sizes of **128/192/256 bits,** the number of rounds are **10/12/14**. AES is an iterated cipher (rather than Feistel cipher) as it processes data as block of 4 columns of 4 bytes and operates on entire data block in every round. Rijndael was designed to have the following characteristics:
The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. In the same way, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and the total key schedule is 44 words for the 128-bit key.

1. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round; these are indicated in above figure.

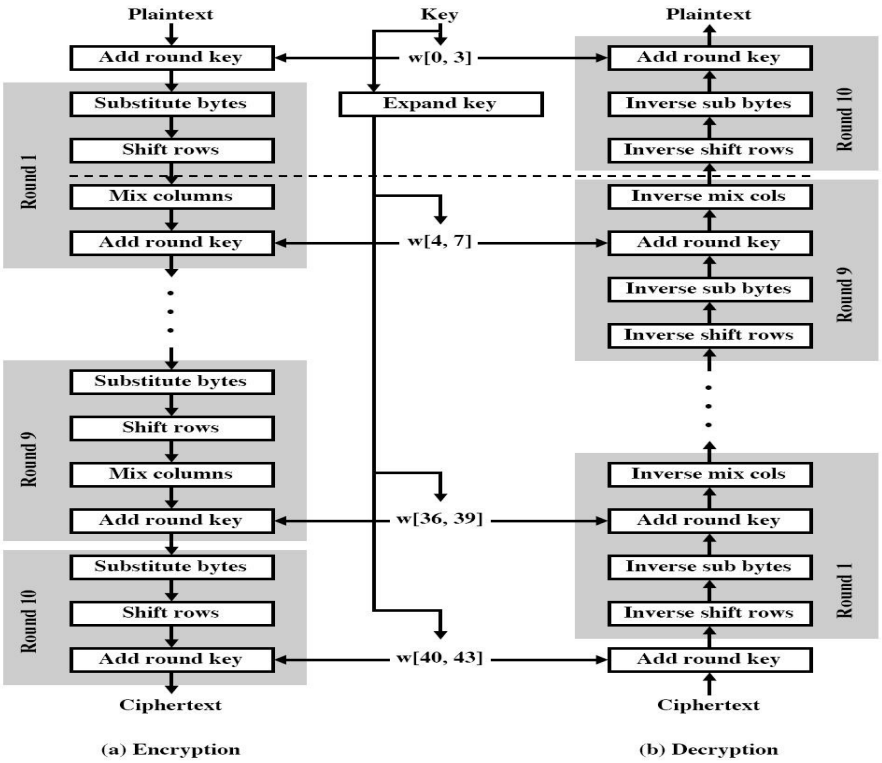
2. Four different stages are used, one of permutation and three of substitution:

I. *Substitute bytes:* Uses an S-box to perform a byte-by-byte substitution of the block II. *ShiftRows:* A simple permutation

III. *MixColumns*: A substitution that makes use of arithmetic over GF(28)
IV. *AddRoundKey:* A simple bitwise XOR of the current block with a portion of the expanded key

3. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. The following figure depicts the structure of a full encryption round.
Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.

4. The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on. This scheme is both efficient and highly secure.



Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm. For the AddRoundKey stage, the inverse is achieved by XORing the same round key to the block, using the result that

7. As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES.

8. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. AES structure figure lays out encryption and decryption going in opposite vertical directions. At each horizontal point (e.g., the dashed line in the figure), State is the same for both encryption and decryption.

9. The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

3.4.Blowfish

Blowfish was developed in 1993 by Bruce Schneier, an independent consultant and cryptographer, and quickly became one of the most popular alternatives to DES.

- Blowfish is easy to implement and it has a high execution speed. It is also a very compact algorithm that can run in less than 5K of memory.

- The key length of Blowfish is variable and can be as long as 448 bits, In practice, 128-bit keys are used. Blow-fish uses 16 rounds.

Blowfish uses S-boxes and the XOR function, similar to DES, but also uses binary addition. Unlike DES, which uses fixed S-boxes, Blowfish uses dynamic S-boxes that are generated as a function of the key.

- In Blowfish, the subkeys and the S-boxes are generated by repeated application of the Blowfish algorithm itself to the key. Blowfish encryption algorithm requires a total of 521 executions to produce the subkeys and S-boxes. As a result, Blowfish is not suitable for applications in which the secret key changes frequently.

Blowfish is one of the most difficult symmetric encryption algorithms so far implemented, because both the subkeys and the S-boxes are produced by a process of repeated applications of Blowfish itself, which thoroughly mangles (crush) the bits and makes cryptanalysis very difficult. So far, no practica l weaknesses have been found in Blowfish cryptanalysis. Blowfish is used in a number of commercial applications.

3.5. Differential and Linear Cryptanalysis

Differential cryptanalysis and Linear cryptanalysis. Both make use of Known plaintext -ciphertext pairs and try to attack the round structure and the S-Boxes. Recent advancements showed that using Differential cryptanalysis, DES can be broken using 2^{47} plaintext-ciphertext pairs and for linear cryptanalysis, the number is even reduced to 2^{41}.
One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis.

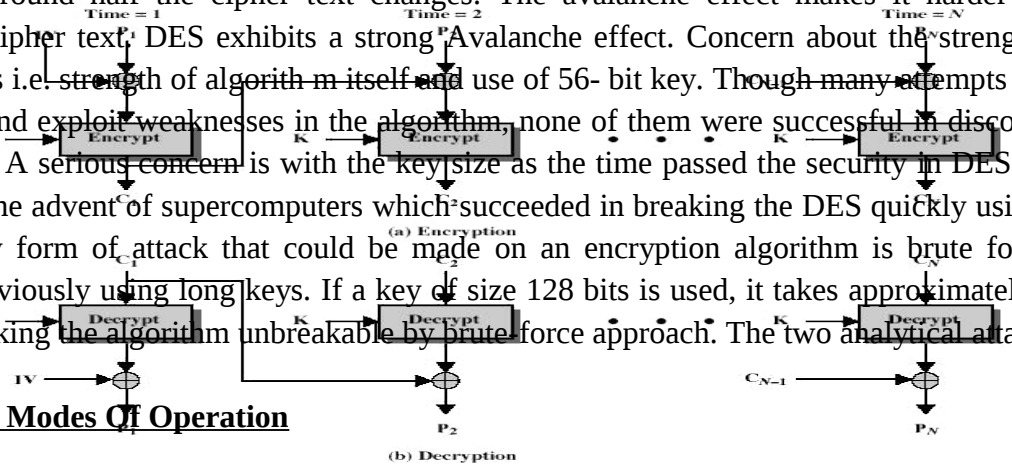
Differential Cryptanalysis

Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 255 complexity The scheme, as reported in [BIHA93], can successfully cryptanalyze DES with an effort on the order of 2^{47} encryptions, requiring 2^{47} chosen plaintexts. Although 2^{47} is certainly significantly less than 2^{55} the need for the adversary to find 2^{47} chosen plaintexts makes this attack of only theoretical interest.

Linear Cryptanalysis

A more recent development is linear cryptanalysis, described in [MATS93]. This attack is based on finding linear approximations to describe the transformations performed in DES. This method can f ind a DES key given 2^{43} known plaintexts, as compared to 2^{47} chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES. So far, little work has been done by other groups to validate the linear cryptanalytic approach.

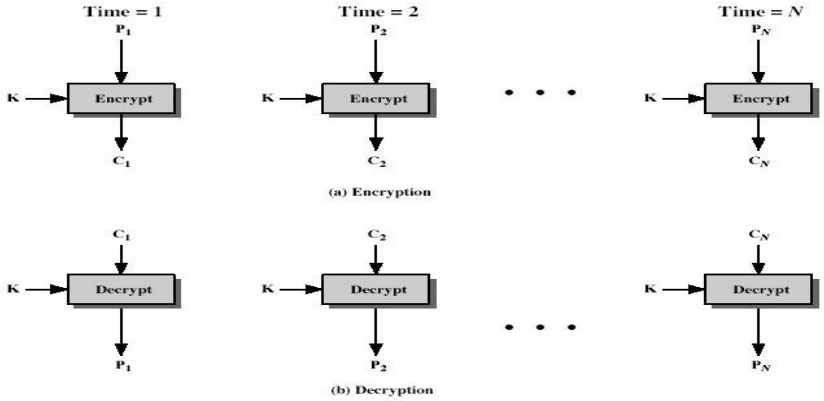
Avalanche Effect: An effect in DES and other secret key ciphers where each small change in plaintext implies that somewhere around half the cipher text changes. The avalanche effect makes it harder to successfully cryptanalyze the cipher text. DES exhibits a strong Avalanche effect. Concern about the strength of DES falls into two categories i.e. strength of algorith m itself and use of 56- bit key. Though many attempts were made over the years to find and exploit weaknesses in the algorithm, none of them were successful in discovering any fatal weakness in DES. A serious concern is with the key size as the time passed the security in DES became getting compromised by the advent of supercomputers which succeeded in breaking the DES quickly using a brute-force attack. If the only form of attack that could be made on an encryption algorithm is brute force, the way of countering it is obviously using long keys. If a key of size 128 bits is used, it takes approximately 1018 years to break the code making the algorithm unbreakable by brute-force approach. The two analytical attacks on DES are



3.6. Block Cipher Modes Of Operation

To apply a block cipher in a variety of applications, four "modes of operation" have been defined by NIST (FIPS

81). The four modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used. As new applications and requirements have appeared, NIST has expanded the list of recommended modes to five in Special Publication 800-38A. These modes are intended for use with any symmetric block cipher, including triple DES and AES. **Electronic Codebook Book (ECB)** The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key. *ECB is the simplest of the modes, and is used when only a single block of info needs to be sent.*



Break the plaintext into 64-bit blocks and encrypt each of them with the same key. The last block should be padded to 64-bit if it is shorter. Same block and same key always yields same cipher block. Each block is a value which is substituted, like a codebook, hence the name Electronic Code Book. Each block is encoded independently of the other blocks. **Ci = DESK1(Pi)** ECB is not appropriate for any quantity of data, since repetitions can be seen, esp. with graphics, and because the blocks can be shuffled/inserted without affecting the en/decryption of each block. Its main use is to send one or a very few blocks, eg a session encryption key.

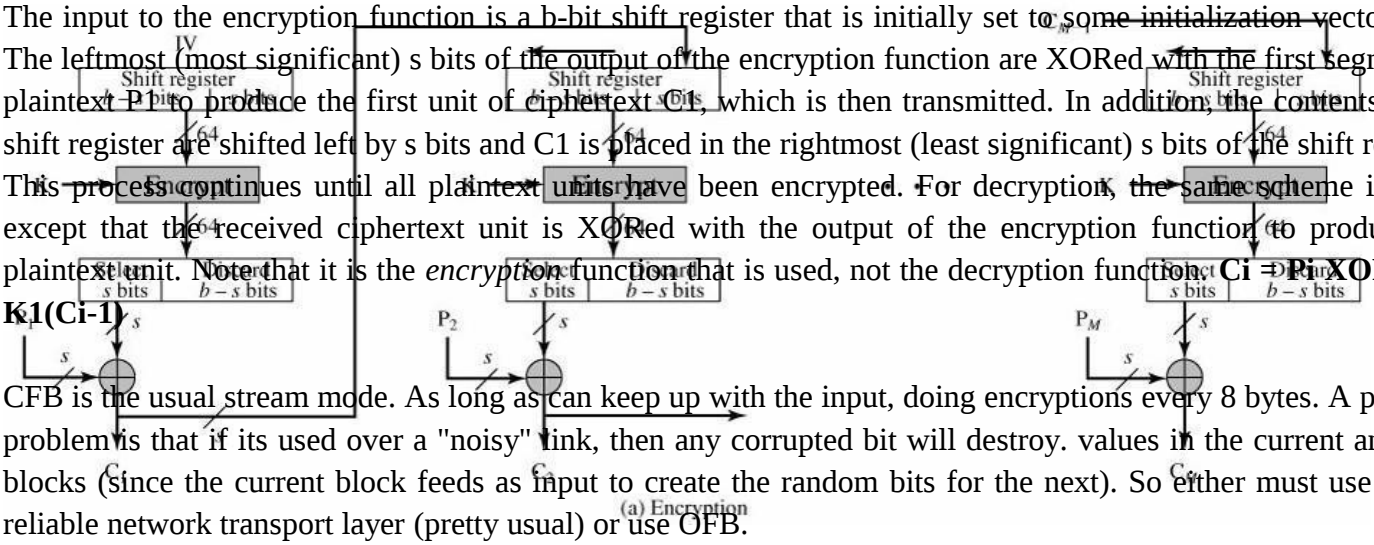
Cipher Block Chaining Mode (CBC) To overcome the problems of repetitions and order independence in ECB, want some way of making the ciphertext dependent on **all** blocks before it. This is what CBC gives us, by combining the previous ciphertext block with the current message block before encrypting. To start the process, use an Initial Value (IV), which is usually well known (often all 0's), or otherwise is sent, ECB encrypted, just before starting CBC use.

All cipher blocks will be chained so that if one is modified, the ciphertext cannot be decrypted correctly. Each plaintext block is XORed with the previous cipher block before encryption, hence the name CBC. The first plaintext block is XORed with an initialization vector IV, which is to be protected securely, (e.g., send it encrypted in ECB mode). **Ci = DESK1(Pi XOR Ci-1)** CBC is the block mode generally used. The chaining provides an avalanche effect, which means the encrypted message cannot be changed or rearranged without totally destroying the subsequent data. However there is the issue of ensuring that the IV is either fixed or sent encrypted in ECB mode to stop attacks on 1st block. **Cipher Feed Back Mode (CFB)**

If the data is only available a bit/byte at a time (eg. terminal session, sensor value etc), then must use some other approach to encrypting it, so as not to delay the info. it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a cipher text output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.

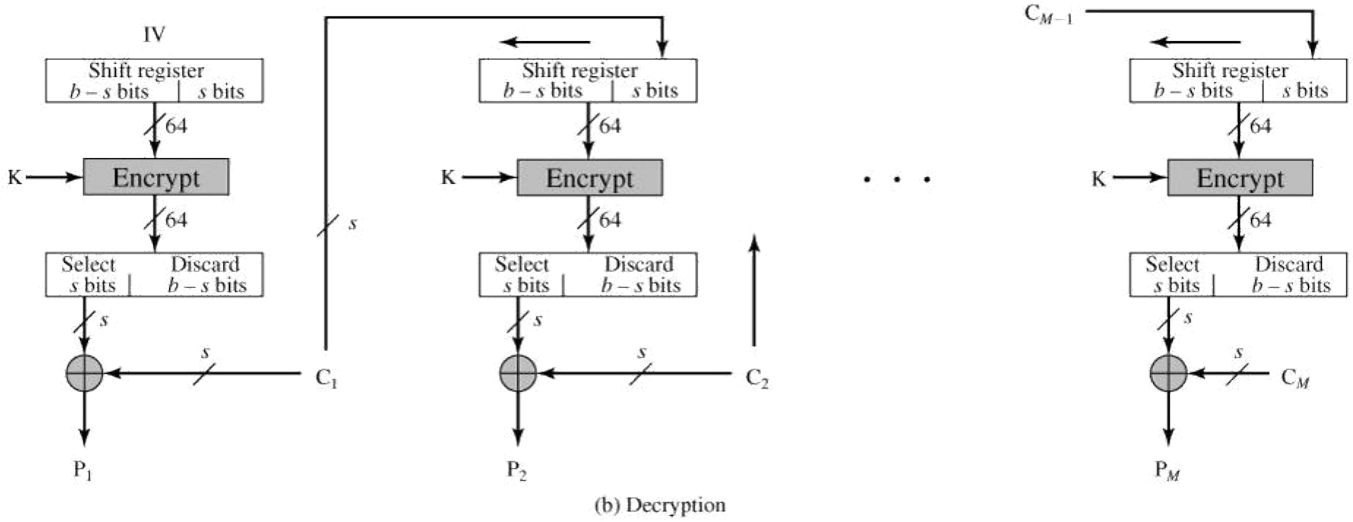
The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of ciphertext C1, which is then transmitted. In addition, the contents of the shift register are shifted left by s bits and C1 is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext units have been encrypted. For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the *encryption* function that is used, not the decryption function. **Ci = Pi XOR DES K1(Ci-1)**



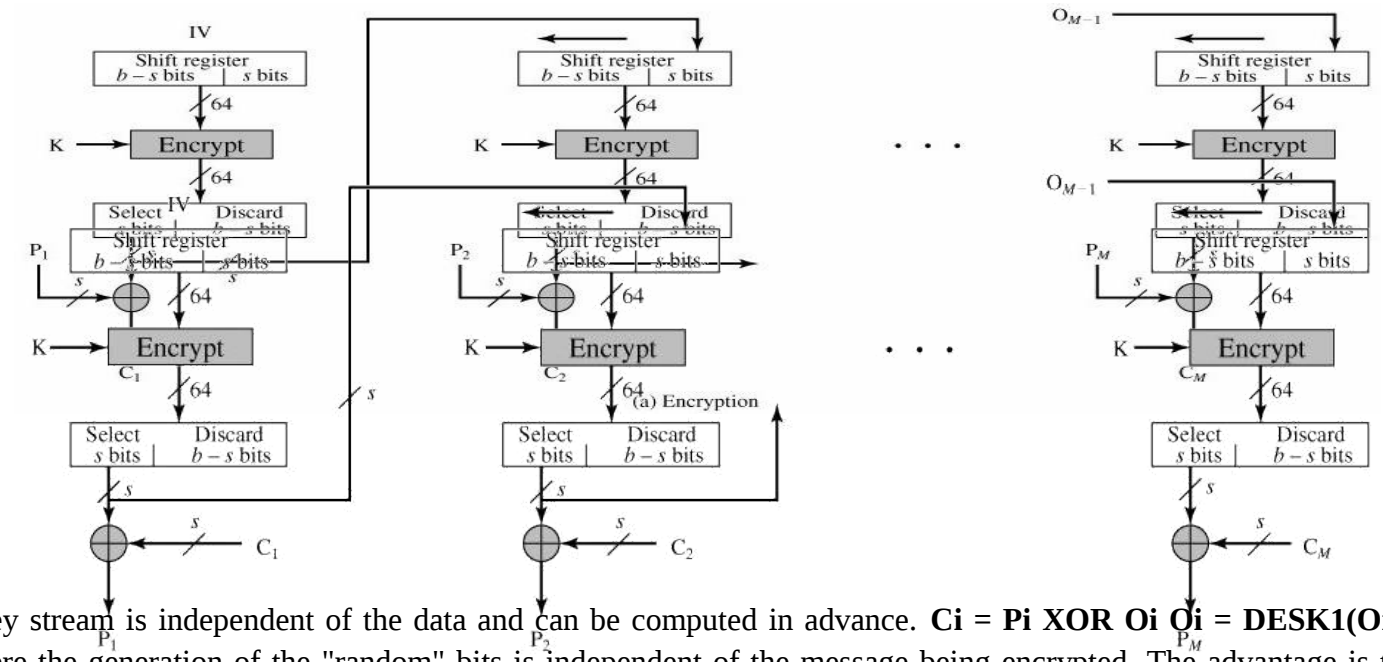
(a) Encryption

CFB is the usual stream mode. As long as can keep up with the input, doing encryptions every 8 bytes. A possible problem is that if its used over a "noisy" link, then any corrupted bit will destroy. values in the current and next blocks (since the current block feeds as input to create the random bits for the next). So either must use over a reliable network transport layer (pretty usual) or use OFB.

Output Feedback Mode (OFB) The output feedback (OFB) mode is similar in structure to that of CFB. It is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register.

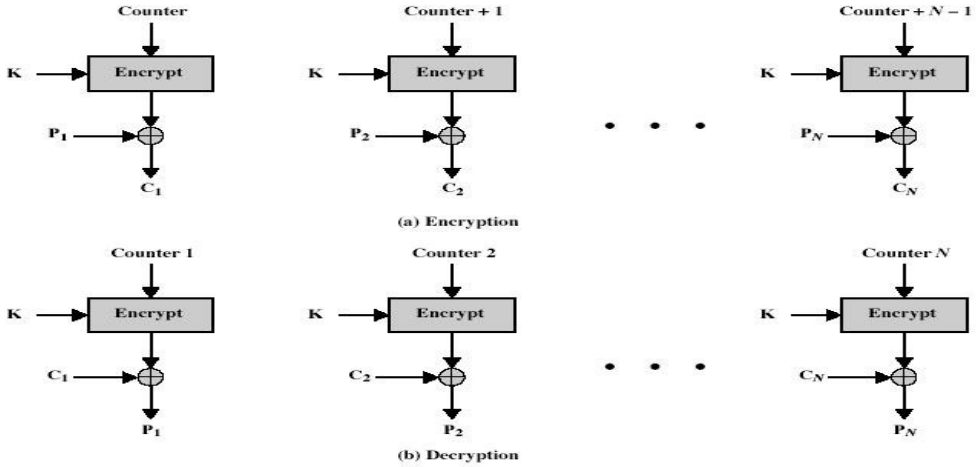


(b) Decryption



Key stream is independent of the data and can be computed in advance. **Ci = Pi XOR Oi Oi = DESK1(Oi-1)** Here the generation of the "random" bits is independent of the message being encrypted. The advantage is that firstly, they can be computed in advance, good for bursty traffic, and secondly, any bit error only affects a single bit. Thus this is good for noisy links (eg satellite TV transmissions etc). The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB.

Counter Mode (CTR) The Counter (CTR) mode is a variant of OFB, but which encrypts a counter value (hence name). Although it was proposed many years before, it has only recently been standardized for use with AES along with the other existing 4 modes. It is being used with applications in ATM (asynchronous transfer mode) network security and IPSec (IP security). All modes of operations except ECB make random access to the file impossible: to access data at the end of the file one has to decrypt everything. Plaintext is not encrypted directly. IV plus a constant is encrypted and the resulting ciphertext is XORed with the plaintext – add 1 to IV in each step.



If the same IV is used twice with the same key, then cryptanalyst may XOR the ciphers to get the XOR of the plaintexts –this could be used in an attack. A counter, equal to the plaintext block size is used. The only

requirement stated in SP 800-38A is that the counter value must be different for each plaintext block that is encrypted. Typically the counter is initialized to some value and then incremented by 1 for each subsequent block. CTR mode has a number of advantages in parallel h/w & s/w efficiency, can preprocess the output values in advance of needing to encrypt, can get random access to encrypted data blocks, and is simple. But like OFB have issue of not reusing the same key + counter value.

3.7. Stream ciphers

Stream Cipher Structure

A *stream cipher* processes the input elements continuously, producing output one element at a time, as it goes along.

A stream cipher encrypts one byte of plaintext at a time, although a stream cipher is designed to operate on one bit at a time or on units larger than a byte at a time. The following Fig 2.7 shows the stream cipher structure.

The input is a key to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently (actually) random. We cannot predict a pseudorandom stream without the knowledge of the input key and it has an apparently random character.

- The output of the generator is called a key stream. It is combined one byte at a time with the plaintext stream using the bitwise XOR operation.

- For ex: if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is:

11001100	plaintext
01101100	key stream
10100000	ciphertext

Decryption requires the same pseudorandom sequence:

10100000	ciphertext
01101100	key stream
11001100	plaintext

The following are the important considerations for stream cipher:

1. The encryption sequence should have a larger period. The longer the period of repeat the more difficult it will be to do cryptanalysis.

2. The key stream should be as random as possible. The more random the key stream, it will be more

difficult for cryptanalysis.

3. The output of the pseudorandom number generator depends on the value of the input key. To protect against brute-force attacks, the key needs to be sufficiently long. A key length of at least 128 bits is

   required.

If the pseudorandom number generator is properly designed, a stream cipher can be as secure as block cipher. The main advantage of stream cipher is that stream ciphers are almost always faster and use far less code than block ciphers.

The advantage of block cipher is that they keys can be reused.

A stream cipher is a better alternative for applications that require encryption/decryption of a stream of data, such as over a data comm'tn channel or a browser/Web link.

3.8. RC4

RC4 was developed in 1994 by Ron Rivest, one of the inventors of the public -key algorithm RSA. RC4 is defined in RFC 2040 and was designed to have the following characteristics:

- **Suitable for hardware or software:** RC4 uses only primitive (old) computational operations commonly found on microprocessors.
- **Fast:** To achieve this, RC4 is a simple algorithm and is word oriented. The basic operations work on full words of data at a time.
- **Adaptable to processors of different word lengths:** The number of bits in a word is a parameter (constraint) of RC4: different word lengths yield different algorithms.
- **Variable number of rounds:** The number of rounds is a second parameter of RC4. This parameter allows a tradeoff between higher speed and higher security.
- **Variable-length key:** The key length is a third parameter of RC4. Again, this allows a tradeoff between speed and security.
- **Simple:** RC4's simple structure is easy to implement and simplifies the task of determining the strength of the algorithm.
- **Low memory requirement:** A low memory requirement makes RC4 suitable for smart cards and other devices with restricted memory.
- **High security:** RC4 is intended (proposed) to provide high security with suitable parameters.

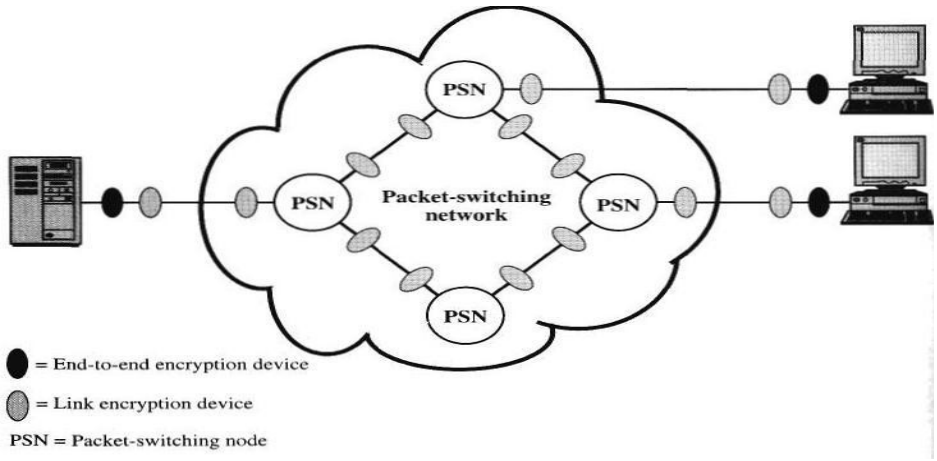
- **Data-dependent rotations:** RC4 incorporates rotations (circular bit shifts) whose amount is data dependent. This appears to strengthen the algorithm against cryptanalysis.

RC4 is used in a number of products from RSA Data Security. Inc.

3.9.Placement of encryption function

Encryption is the most powerful, and most common, approach to counter the threats to n/w security. When using encryption, we need to decide what to encrypt and where the encryption gear (equipment or mechanism) should be located.

- There are two fundamental alternatives: link encryption and end-to-end encryption: these are shown in Fig 2.11 using a packet-switching n/w.



2.11: Encryption across a Packet-Switching N/w

With **link encryption**, each vulnerable (susceptible) comm'tns link is equipped on both ends with an encryption device. Thus, all traffic over all comm'tns links is secured. Although this requires a lot of encryption devices in a large network, it provides a high level of security.

- One disadvantage of this approach is that the message must be decrypted each time it enters a packet switch: this is b'coz the switch must read the address (virtual circuit num) in the packet header to route the packet. Thus, the message is vulnerable (at risk) at each switch. If this is a public packet-switching n/w, the user has no control over the security of the nodes.

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, is then transmitted without any alteration across the n/w to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data.

- This approach seems to provide secure transmission against attacks on the n/w links or switches. But, still there is a weak spot.

Consider the following situation. A host connects to an X.25 packet -switching n/w, sets up a virtual circuit to another host, and is prepared to transfer data to that other host using end-to-end encryption.

- Data is transmitted over such a n/w in the form of packets, consisting 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. But, this will not work b'coz, only the other host can perform the decryption. The packet-switching node will receive an encrypted packet and it will be unable to read the header. Therefore, it will not be able to route the packet.

- So the host may only encrypt the user data portion of the packet and must leave the header in the clear, so that it can be read by the n/w.

Thus, with end-to-end encryption, the user data is secure. But, the traffic pattern is not, b'coz, packet headers are transmitted in the clear.

- So to achieve greater security, both link and end -to-end encryptions are needed, shown in Fig 2.11.

To summarize, when both techniques are used, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link encryption key. As the packet traverses (pass thru) the n/w, each switch decrypts the packet using a link

encryption key to read the header and then encrypts the entire packet again for s ending it out on the next link.

- Now the entire packet is secure except when the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

3.10. KEY DISTRIBUTION

- For symmetric encryption to work, the two parties who are doing a secure exchng must have the same key, and that key must be protected from access by others.

- Therefore, the strength of any cryptographic system rests with the key distribution technique .

- **Key Distribution** refers to the means of delivering a key to two parties that wish to exchange data, without allowing others to see the key.

1. Key distribution can be achieved in a number of ways. For two parties A and B, A key can be selected by A and physically delivered to B.

2. A third party can select the key and physically deliver it to A and B.

3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.

4. If A and B each have an encrypted connection to a third party C, C can deliver a key on the

   encrypted links to A and B.

- Options 1 and 2 call for manual delivery of a key. This is a reasonable requirement for link encryption, b'coz each link encryption device is only going to exchange data with its partner on the other end of the link.

- Option 3 can be used for either link encryption or end-to-end encryption, but if an attacker succeeds in gaining access to one key, then all subsequent keys are revealed.

- Even if frequent changes are made to the link encryption keys , these should be done manually.

- To provide keys for end-to-end encryption, option 4 is preferable.

-

Fig 2.12 Automatic Key Distribution for Connection-Oriented Protocol

- Fig 2.12 shows an implementation that satisfies option 4 for end -to-end encryption. Link encryption is ignored in the fig. This can be added, or not, as required. For this scheme, two kinds of keys are identified:

  - **Session key:** When two end systems (hosts, terminals, etc.) wish to commu nicate, they establish a logical connection (e.g., virtual circuit).

- For the duration of that logical connection, all user data is encrypted with a one-time session key.
- At the conclusion of the session, or connection, the session key is destroyed.

- **Permanent key:** A permanent key is a key used between entities for the purpose of distributing session keys.

- The configuration consists of the following elements:

  - **Key distribution center:** The key distribution center (KDC) determines which systems are allowed to communicate with each other.
    - When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.
  - **Security service Module (SSM)*:*** The SSM performs end-to-end encryption and obtains session keys on behalf of its host or terminal or users.

The steps involved in establishing a connection are shown in Figure 2.12.

- **Step 1:** When one host wishes to set up a connection to another host, it transmits a connection-request packet.
- **Step 2:** The SSM saves that packet and applies to the KDC for permission, to establish the connection.
- **Step 3:** The communication between the SSM and the KDC is encrypted using a master key shared only by the SSM and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM.
- **Step 4:** The requesting SSM can now release the connection request packet, and a connection is set up, b/w the two end systems. All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.

The automated key distribution approach provides the flexibility and dynamic characteristics. It allows a num of terminal users to access a num of hosts and for the hosts to exchange data with each other.

4.Asymmetric Key Ciphers

4.1. Public key cryptography principles

Public key encryption is as important as conventional encryption. Public key encryption is used in message authentication and key distribution.

Public-Key Encryption Structure

- Public-key encryption was first proposed by Diffie and Hellman in 1976.

- Public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns.

- Public-key cryptography is asymmetric, i.e., it involves the use of two separate keys.
- The use of two keys has a strong effect in the areas of confidentiality, key distribution, and authentication.
- There are a few common misconceptions about public-key encryption.

- One is it is believed that public-key encryption is more secure from cryptanalysis than conventional encryption. But the fact is, the security of any encryption scheme depends on (1) the length of the key and

  (2) the computational work involved in breaking a cipher. It does not depend on whether we are using either conventional or public-key encryption that makes one superior (better) to another from resisting cryptanalysis.

- A second misconception is that public-key encryption is a general-purpose technique that made conventional encryption obsolete (out of date). On the contrary, because of the computational overhead of current public-key encryption schemes, there are no expected chances that conventional encryption will be discarded.

- Finally, there is a feeling that key distribution is trivial (simple) when using public-key encryption, compared to the process involved with key distribution centers for conventional encryption. In fact, some form of protocol is needed, which frequently involves a central agent, and the procedures involved are not simpler or more efficient than those required for conventional encryption.

A public-key encryption scheme has six ingredients (Fig 3.1a):

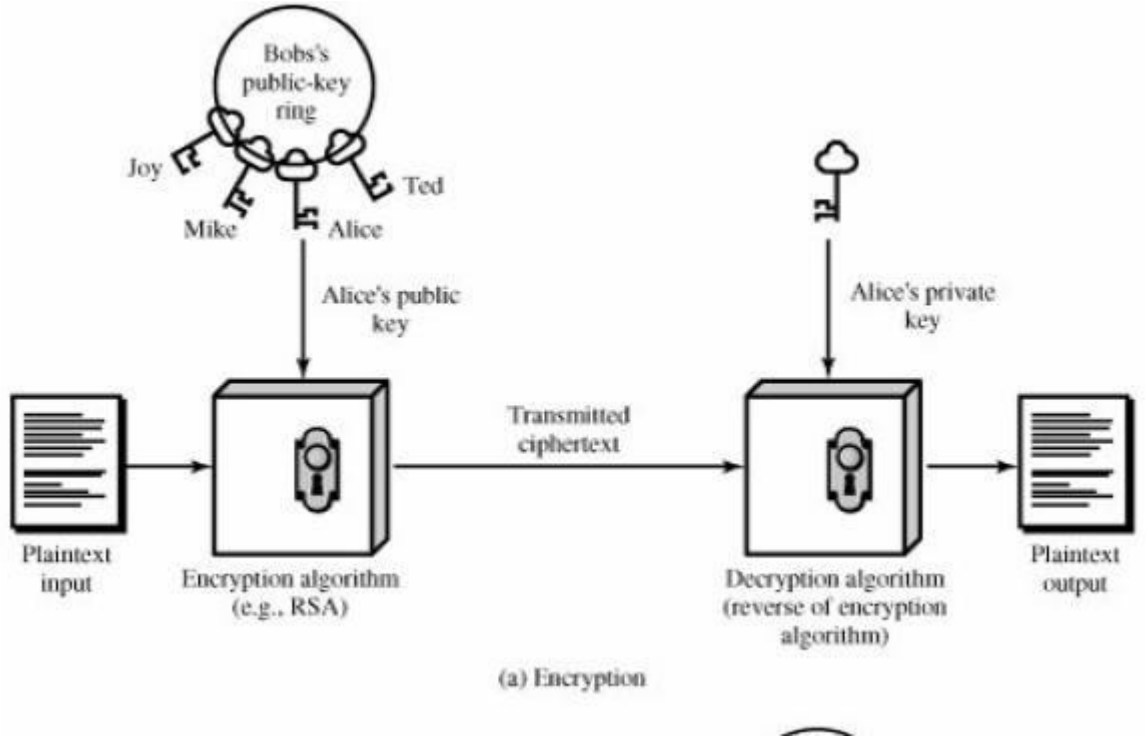


Fig 3.1a: Public-Key Cryptography (Encryption)

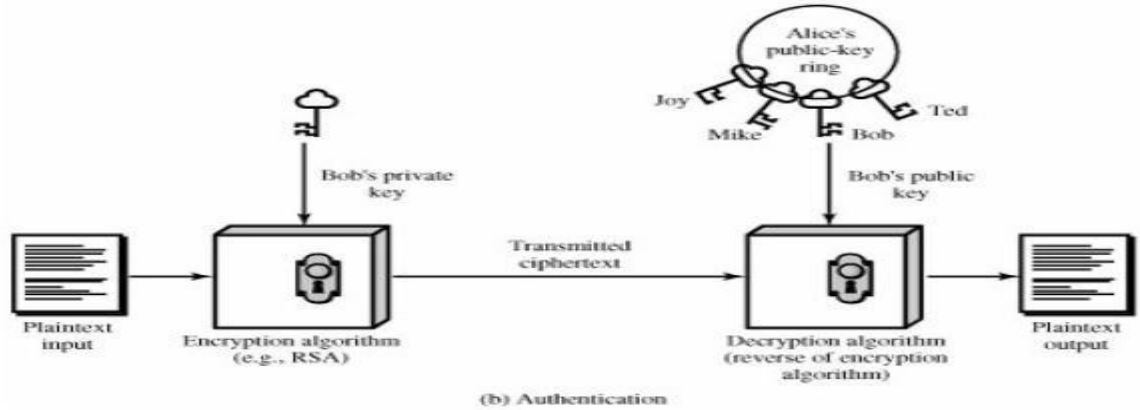


Fig Public-Key Cryptography (Authentication)

Plaintext: This is the readable message or data that is fed into the algorithm as input.
Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.

Public and private key: This is a pair of keys that are selected so that if one is used for encryption, the other is

used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.

Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.

Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

As the names suggest, the public key is made public for others to use, while the private key is known only to its owner. A general-purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption.

The following are the essential steps:

Each user generates a pair of keys to be used for the encryption and decryption of messages.

Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As shown in fig 3.7a, each user maintains a collection of public keys obtained from others.

If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.

When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

With this approach, all participants will have access to public keys, and private keys are generated locally by each participant and they are not distributed.

The incoming comm'tion is secure as long as a user protects his or her private key. At any time, a user can change the private key and publish the companion public key to replace the old public key.

The key used in conventional encryption is referred as a secret key.

The two keys used for public-key encryption are referred to as the public key and the private key. The private key is always kept secret, but it is referred as a private key instead of a secret key to avoid confusion with conventional encryption.

Applications for Public-Key Cryptosystems

Public-key systems are use of a cryptographic type of algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function.

The use of public-key cryptosystems are classified into three categories:
Encryption/decryption: The sender encrypts a message with the recipient's public key.

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

Key exchange: Two sides cooperate to exchange a session key by using different approaches involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications.

Table 3.1 shows the applications supported by the algorithms, RSA and Diffie Hellman, the Digital Signature Standard (DSS) and elliptic-curve cryptography.

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

Table 3.1: Applications for Public-Key Cryptosystems Requirements for Public-Key Cryptography

The cryptosystem shown in Figure 3.1 depends on a cryptographic algorithm based on two related keys. This system was suggested by Diffie and Hellman, but they did not give show any demonstration to the existence of such algorithms.
But, they have give some conditions which need to be fulfilled by such algorithms:

It is computationally easy for a party B to generate a pair (public key PUb, private key PRb).

It is computationally easy for a sender A, knSowing the public key and the message to be encrypted, M. to generate the corresponding ciphertext:
C = E(PUb, M)

It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

M = D(PRb, C) = D[PRb, E(PUb, M)].

It is computationally infeasible for an opponent, knowing the public key, PUb, to determine the private key, PRb.

It is computationally infeasible for an opponent, knowing the public key, PUb, and a ciphertext, C, to recover the original message, M.
Either of the two related keys can be used for encryption, with the other used for decryption.

But the sixth requirement is not necessary for all public-key applications.
M = D[PUb, E(PRb, M)] = D[PRb, E(PUb, M)]

PUBLIC KEY CRYPTOGRAPHY ALGORITHMS
The two most widely used public-key algorithms are RSA and Diffie-Hellman.

4.2. RSA

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 all 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-key. 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 encryption 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) Its 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 any 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,

p=17 & q=11
n = pq
=17×11=187

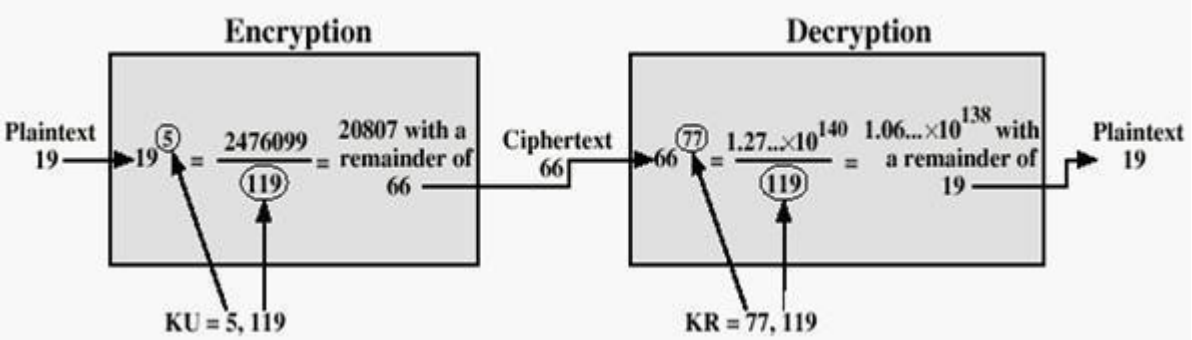
Compute ø(n)=(p–1)(q-1)=16×10=160 Select e *:* gcd(e,160)=1; choose e=7

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

Another example is:



For RSA key generation,

– determine two primes at random - p, q
– select either e or d and compute the other

– means must be sufficiently large
– typically guess and use probabilistic test
onents e, d are inverses, so use Inverse algorithm to compute the other

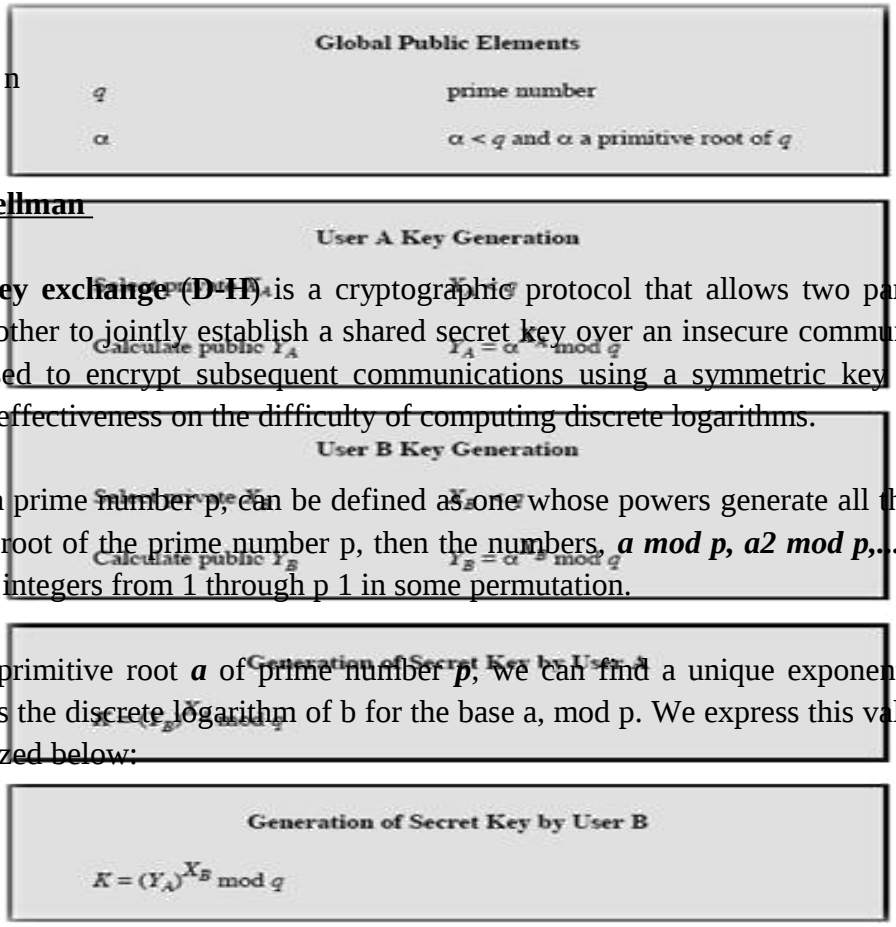
Security of RSA There are three main approaches of attacking RSA algorithm. **Brute force key search** (infeasible given size of numbers) As explained before, involves trying all possible private keys. Best defence is using large keys. **Mathematical attacks** (based on difficulty of computing ø(N), 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. **Timing attacks** (on running of decryption) These attacks involve determination of a private key by keeping track of how long a computer takes to decipher a message (ciphertext-only attack) –this is essentially an attack on the fast exponentiation algorithm but can be adapted for any other algorithm. Though these attacks are a quite serious threat, there are some simple countermeasures that can be used. They are explained below: *Constant exponentiation time*:- Ensure that all exponentiations take the same time before returning a result: degrade performance of the algorithm. *Random Delay*:- A random delay can be added to exponentiation algorithm to confuse the timing attack. If there is not enough noise added by defenders, the attackers can succeed. *Blinding:*- Multiply the ciphertext by a **random** number before performing exponentiation –in this way the attacker does not know the input to the exponentiation algorithm. RSA Data Security incorporates a blinding feature into some of its products. The private-key operation M = Cd mod n is implemented as follows:

Generate a secret random number r between 0 and n-1

-1mod n



4.3. **Diffie – Hellman**

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. The D-H algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

First, a primitive root of a prime number p, can be defined 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 .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 ≡ c (mod p) and 0 ≤ 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 ≤ c < p) and a modulus p, calculate the exponent b such that ab ≡ c (mod 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.

ey

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.

4.4. 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 Neal 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 size, thereby reducing the processing overhead. **Elliptic Curve over GF(p)** ***4a3 + 27b2 0 (mod p).*** An

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.

then P +Q = (x3 ,y3) where ***x3 =*** ***2 - x1 - x2 y3 = (x1 - x3) - y1 and*** ***= (y2-y1)/(x2-x1) if P ≠ Q*** =

(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 + xy***

= 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 addition is 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. **ECC Key Exchange** Pick a prime number p= 2180 and elliptic curve parameters a and b for the equation ***y2 x3 + ax + b (mod 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 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 ciphertext, 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+*k*(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.

Hence for similar security ECC offers significant computational advantages.

Applications of ECC:

necessary for our current cryptosystems

4.5. 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.

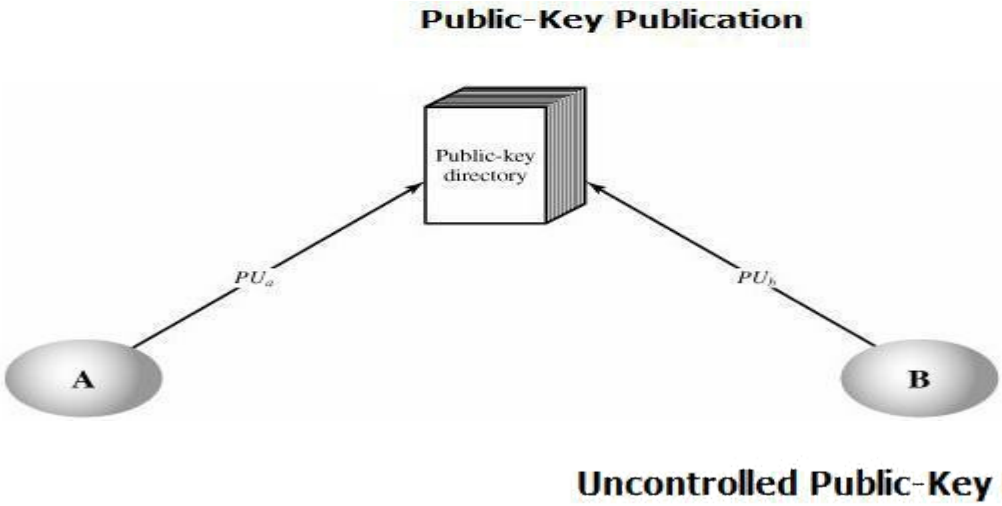
Distribution of Public Keys The most general schemes for distribution of public keys are given below

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.

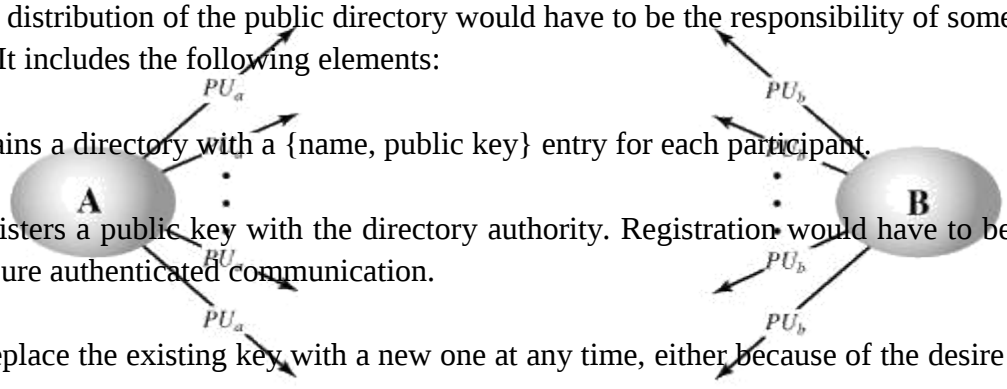
Though this approach seems convenient, it has a major drawback. Anyone can forge such a public announcement. Some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until the time when A discovers about the forgery and alerts other participants,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

PUBLICLY AVAILABLE DIRECTORY



Uncontrolled Public-Key Distribution

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:



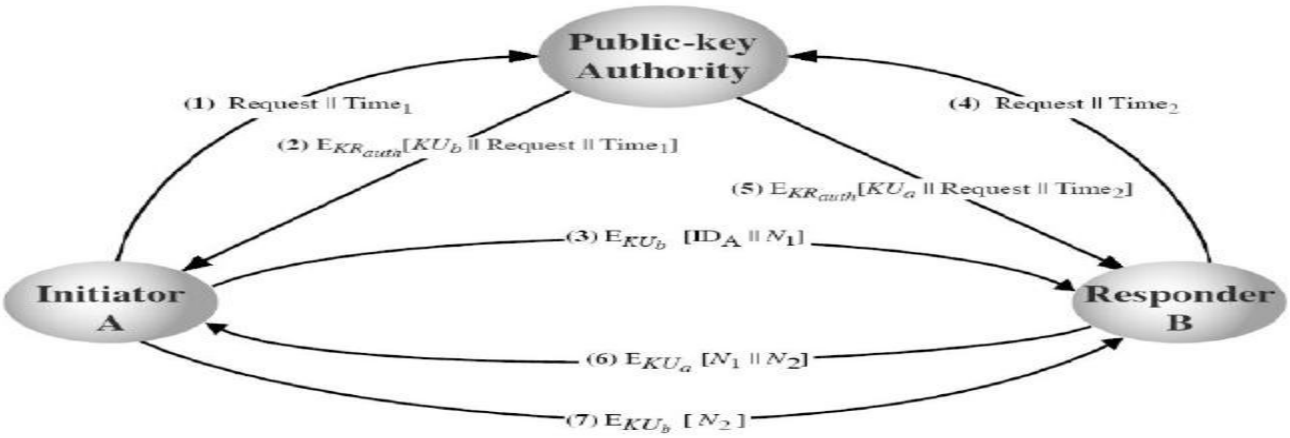
1. The authority maintains a directory with a {name, public key} entry for each participant.

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 vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Or else, the adversary may tamper 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 time stamped message to the central authority with a request for Bob's public key (the time stamp is to mark the moment of The authority sends back a message encrypted with its private key (for authentication) –message contains Bob's public key 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 N1 (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 name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated 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.

*** * * * * * * * * * * END OF UNIT II * * * * * * * * * * * ***

UNIT III

MESSAGE AUTHENTICATION REQUIREMENT

ATTACKS ON COMMUNICATION OVER NETWORKS

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity and also fraudulent acknowledgments
of message receipt or non receipt by someone other than the message recipient.
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.
8. **Destination repudiation:** Denial of receipt of message by destination.

- Items (3) through (6) in the foregoing list are generally regarded as message authentication.
- Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness.
- A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

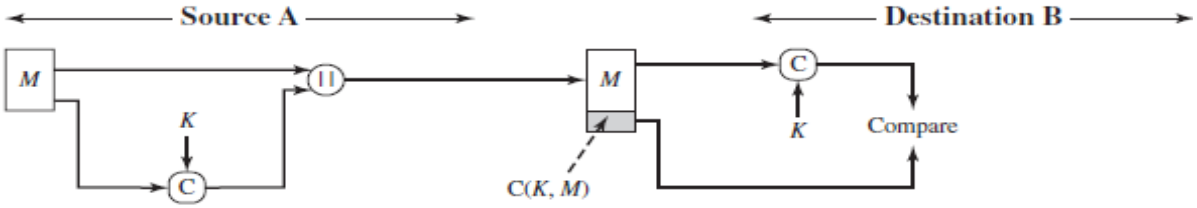
Message Authentication Functions

- ✓ Message authentication has two levels of functionality.
- ✓ At the lower level, a function that produces an authenticator, a value to be used to authenticate a message.
- ✓ In a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.
- ✓ Different types of functions that may be used to produce an authenticator are grouped into three classes

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

Message Authentication Code

- An authentication technique that involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or MAC, that is appended to the message.
- Assumes that two communicating parties, say A and B, share a common secret key *K*.
- MAC is calculated as **MAC = C(*K*, *M*)** where
  M = input message,C = MAC function,K = shared secret key,MAC = message authentication code. MAC(*K*, *M*) is the fixed-length authenticator, sometimes called a **tag**.
- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC.

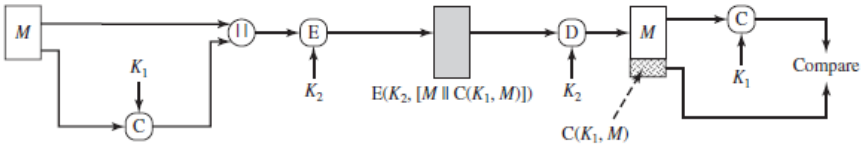


(a) Message authentication

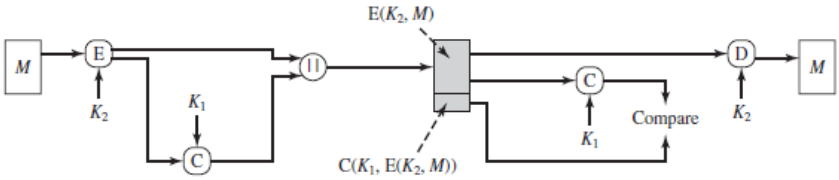
- If the received MAC matches the calculated MAC
  1. *The receiver is assured that the message has not been altered.*
  2. *The receiver is assured that the message is from the alleged sender*
  3. *If the message includes a sequence number then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number*.

S.NO	MAC	ENCRYPTION
1	Need not be reversible	Decryption need to be reversible
2	Many to one function	One to one function.
3	Less vulnerable to be broken	More vulnerable than MAC

MESSAGE AUTHENTICATION AND CONFIDENTIALITY



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



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

- Perform message encryption either after the MAC algorithm.
- Two separate keys are needed, each of which is shared by the sender and the receiver.
- It is preferable to tie the authentication directly to the plaintext.
- Assuming that an opponent knows the MAC function but does not know *K*. Then the MAC function should satisfy the following requirements.
  1. *If an opponent observes M and MAC (K, M), it should be computationally infeasible for the opponent to construct a message M 'such that MAC (K, M') = MAC (K, M)*
  2. *MAC (K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M ', the probability that MAC(K, M) = MAC(K, M ') is 2^{-n}, where n is the number of bits in the tag.*
  3. *Let M ' be equal to some known transformation on M. That is, M '= f(M). For example, f may involve inverting one or more specific bits. In that case,*

     Pr [MAC(K, M) = MAC(K, M ')] = 2^{-n}

Security of Macs

- Attacks on MACs are grouped into two categories: brute-force attacks and cryptanalysis.
- <u>BRUTE FORCE ATTACKS</u>
- More difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs.
- The desired security property of a MAC algorithm, can be expressed as follows.
  - *Computation resistance: Given one or more text-MAC pairs [xi, MAC(K, xi)], it is computationally infeasible to compute any text-MAC pair [x, MAC(K, x)] for any new input x ≠xi.*
- There are two lines of attack possible: attack the key space and attack the MAC value.
- Attacking KEY SPACE
  - ✓ If an attacker can determine the MAC key, it is possible to generate a valid MAC value for any input *x*.

- ✓ The key size is *k* bits and that the attacker has one known text-tag pair. Then the attacker computes the *n*-bit tag on the known text for all possible keys.
- ✓ At least one key is guaranteed to produce the correct tag.
- ✓ This phase of the attack takes a level of effort proportional to $2^{k,}$, one operation for each of the 2^k possible key values.,
- ✓ Because the MAC is a many-to-one mapping, there may be other keys that produce the correct value.
- ✓ If more than one key is found to produce the correct value, additional text-tag pairs must be tested.
- ✓ The level of effort drops off rapidly with each additional text-MAC pair and that the overall level of effort is roughly 2^k

- ATTACKING MAC VALUE
  - ✓ Generate a valid tag for a given message or to find a message that matches a given tag.
  - ✓ The level of effort is comparable to that for attacking the one-way or weak collision-resistant property of a hash code, or $2^{n.}$
  - ✓ The attack cannot be conducted off line without further input; the attacker will require chosen text-tag pairs or knowledge of the key.
  - ✓ The level of effort for brute-force attack on a MAC algorithm can be expressed as min (2^k, 2^n).
- Cryptanalytic attacks
  - ✓ Cryptanalytic attacks on MAC algorithms seek to exploit some property of the algorithm to perform some attack
  - ✓ other than an exhaustive search.
  - ✓ The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a bruteforce attack.
  - ✓ An ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

HMAC

- ✓ MAC derived from a cryptographic hash function.
- ✓ The motivations for this interest are

  1. Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES.

  2. Library code for cryptographic hash functions is widely available.
- ✓ HMAC stands for Hash-based MAC. It works by using an **underlying hash function** over a message and a key.
- ✓ Any hash function could be used with HMAC, although more secure hashing functions are preferable. Commonly used hash functions are MD5 and SHA-1.

- ✓ As computers become more and more powerful, increasingly complex hash functions will probably be used.
- ✓ Speed is the main reason. Hash functions are much faster than block ciphers such as DES and AES in software implementation
- ✓ Another advantage is that they are freely available, and are not subject to the export restriction rules of the USA and other countries.
- ✓ However, HMAC, as a cryptographic mechanism, is repudiatable. That is, Bob cannot demonstrate that data really came from Alice -- both a sender and a receiver can generate an exactly same HMAC output (so Bob could have made the data himself). This is unlike digital signatures which only the sender can generate.
- ✓ You use HMAC whenever you want **integrity** of the data maintained (and authenticity)
- ✓ The key is part of the HMAC, since it is a shared secret known between 2 parties only and only they can create the HMAC and no one else. (Ensures **authenticity**)
- ✓ Length extension attacks are **not possible** on HMAC. MAC's on the other hand simply appends key to the message, which is susceptible to it. HMAC was introduced to overcome this attack on MAC's.

HMAC Design Objectives

- To use, without modifications, available hash functions.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function

HMAC Algorithm

- *H* = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
- *IV* = initial value input to hash function
- *M* = message input to HMAC (including the padding specified in the embedded hash function)
- *Yi* = *i* th block of M, 0 … *i* … (*L* - 1)
- *L* = number of blocks in *M*
- *b* = number of bits in a block
- *n* = length of hash code produced by embedded hash function
- *K* = secret key; recommended length is Ú *n*; if key length is greater than *b*, the key is input to the hash function to produce an *n*-bit key
- *K*+ = *K* padded with zeros on the left so that the result is *b* bits in length

- ipad = 00110110 (36 in hexadecimal) repeated *b*/8 times
- opad = 01011100 (5C in hexadecimal) repeated *b*/8 times

Then HMAC can be expressed as

HMAC(*K*, *M*) = H[(*K*+ xor opad) || H[(*K*+ xor ipad) || *M*]]

We can describe the algorithm as follows.

1. *Append zeros to the left end of K to create a b-bit string K+ (e.g., if K is of length 160 bits and b = 512, then K will be appended with 44 zeroes).*
2. *XOR (bitwise exclusive-OR) K+ with ipad to produce the b-bit block Si.*
3. *Append M to Si.*
4. *Apply H to the stream generated in step 3.*
5. *XOR K+ with opad to produce the b-bit block So.*
6. *Append the hash result from step 4 to So.*
7. *Apply H to the stream generated in step 6 and output the result.*

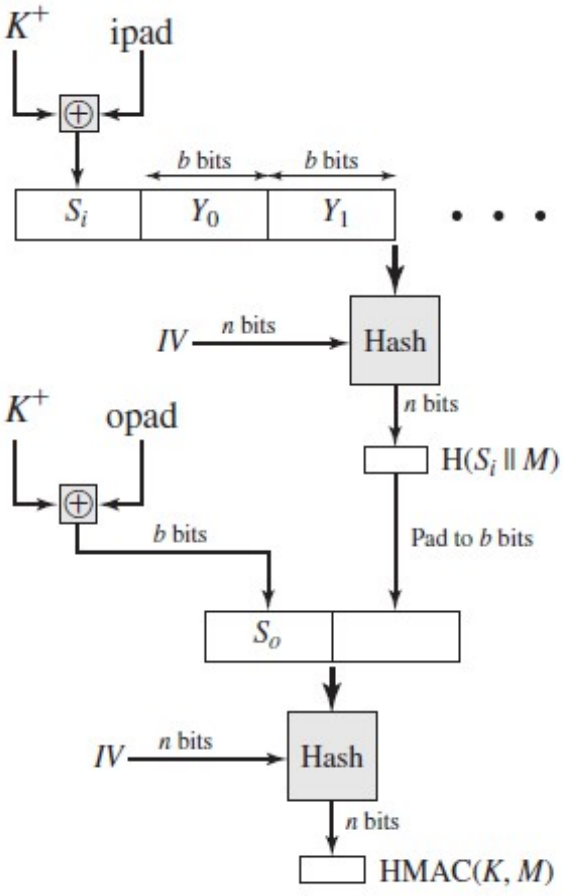


Figure 12.5 HMAC Structure

- XOR with ipad and opad results in flipping one-half of the bits of *K*.

- By passing *Si* and *So* through the compression function of the hash algorithm, we pseudo randomly generated two keys from *K*.
- A more efficient implementation is possible if two quantities are pre computed

f(*IV*, (*K* + XOR ipad))
f(*IV*, (*K* + XOR opad))

- where f(cv, block) is the compression function for the hash function, which takes as arguments a chaining variable of *n* bits and a block of *b* bits and produces a chaining variable of *n* bits.
- These quantities only need to be computed initially and every time the key changes.
- The pre computed quantities substitute for the initial value (*IV*) in the hash function.
- This more efficient implementation is especially worthwhile if most of the messages for which a MAC is computed are short.

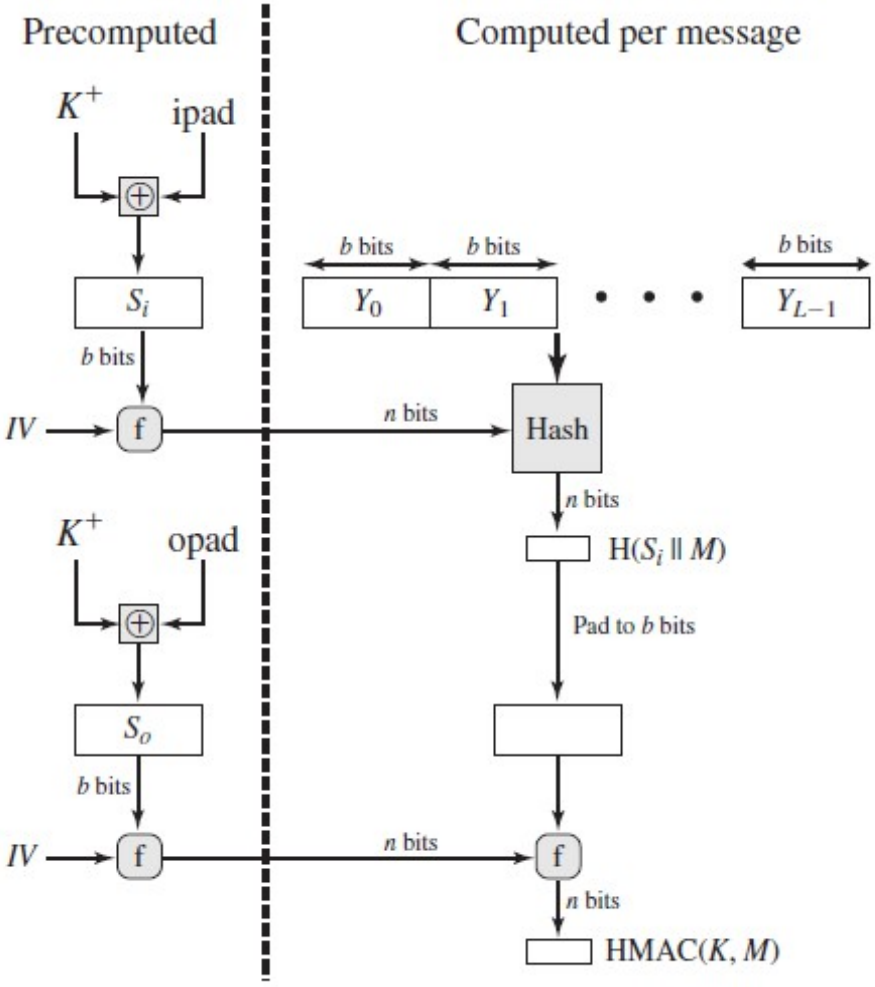


Figure 12.6 Efficient Implementation of HMAC

Security of HMAC

- The security of any MAC function based on an embedded hash function depends in some way on the cryptographic strength of the underlying hash function.

- The security of a MAC function is generally expressed in terms of the probability of successful forgery with a given amount of time spent by the forger and a given number of message-tag pairs created with the same key.
- The probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function.
    - ✓ The attacker is able to compute an output of the compression function even with an *IV* that is random, secret, and unknown to the attacker.
    - ✓ The attacker finds collisions in the hash function even when the *IV* is random and secret.

- In the first attack, the compression function is equivalent to the hash function applied to a message consisting of a single *b*-bit block. For this attack,the *IV* of the hash function is replaced by a secret, random value of *n* bits.
- An attack on this hash function requires either a brute-force attack on the key, which is a level of effort on the order of 2^n, or a birthday attack, which is a special case of the second attack.
- In the second attack, the attacker is looking for two messages *M* and *M* ‘that produce the same hash: H(*M*) = H(*M* ‘). This requires a level of effort of $2^{n/2}$ for a hash length of *n*.
- when attacking HMAC, the attacker cannot generate message/ code pairs off line because the attacker does not know *K*. Therefore, the attacker must observe a sequence of messages generated by HMAC under the same key and perform the attack on these known messages.

CMAC **Cipher-based Message Authentication Code**

- The message is an integer multiple *n* of the cipher block length *b.*
- For AES, *b* = 128, and for triple DES, *b* = 64. The message is divided into *n* blocks (*M*1, *M*2…..*Mn*).
- The algorithm makes use of a *k*-bit encryption key *K* and a *b*-bit constant, *K*1. For AES, the key size *k* is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits.

$C1 = E(K, M1)$
$C2 = E(K, [M2 \text{ XOR } C1])$
$C3 = E(K, [M3 \text{ XOR } C2])$
$Cn = E(K, [Mn \text{ XOR } Cn\text{-}1 \text{ XOR } K1])$
$T = \text{MSB}Tlen(Cn)$
where
T = message authentication code, also referred to as the tag
Tlen = bit length of T

MSB$_s(X)$ = the s leftmost bits of the bit string X.

- If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b.
- The CMAC operation then proceeds as before, except that a different b-bit key K_2 is used instead of K_1.
- The two b-bit keys are derived from the k-bit encryption key as follows.

$$L = E(K, 0^b)$$
$$K1 = L \cdot x$$
$$K2 = L \cdot x^2 = (L \cdot x) \cdot x$$

- where multiplication (.) is done in the finite field GF(2^b) and x and x^2 are first and second-order polynomials that are elements of GF(2^b). Thus, the binary representation of x consists of b - 2 zeros followed by 10; the binary representation of x^2 consists of b - 3 zeros followed by 100.
- The finite field is defined with respect to an irreducible polynomial that is lexicographically first among
  all such polynomials with the minimum possible number of nonzero terms.
- For the two approved block sizes, the polynomials are $x^{64} + x^4 + x^3 + x + 1$ and $x^{128} + x^7 + x^2 + x + 1$.
- To generate $K1$ and $K2$, the block cipher is applied to the block that consists entirely of 0 bits. The first sub key is derived from the resulting cipher text by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size.
- The second sub key is derived in the same manner from the first sub key.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +                Algorithm Generate_Subkey                  +
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  +                                                           +
  +  Input    : K (128-bit key)                              +
  +  Output   : K1 (128-bit first subkey)                    +
  +           K2 (128-bit second subkey)                     +
  +-----------------------------------------------------------+
  +                                                           +
  +  Constants: const_Zero is 0x00000000000000000000000000000000     +
  +           const_Rb   is 0x00000000000000000000000000000087     +
  +  Variables: L          for output of AES-128 applied to 0^128    +
  +                                                           +
  +  Step 1.  L := AES-128(K, const_Zero);                    +
  +  Step 2.  if MSB(L) is equal to 0                         +
  +          then    K1 := L << 1;                           +
```

```
+          else    K1 := (L << 1) XOR const_Rb;                +
+   Step 3.  if MSB(K1) is equal to 0                          +
+          then    K2 := K1 << 1;                              +
+          else    K2 := (K1 << 1) XOR const_Rb;               +
+   Step 4.  return K1, K2;                                    +
+                                                              +
```

DIGITAL SIGNATURES

A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity).

A digital signature scheme typically consists of three algorithms;

- A *key generation* algorithm that selects a *private key* uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding *public key*.
- A *signing* algorithm that, given a message and a private key, produces a signature.
- A *signature verifying* algorithm that, given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required.

First, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key.

Secondly, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key.

A digital signature is an authentication mechanism that enables the creator of the message to attach a code that acts as a signature.

PROPERTIES

The digital signature must have the following properties:
• It must verify the author and the date and time of the signature.
• It must authenticate the contents at the time of the signature.
• It must be verifiable by third parties, to resolve disputes.
Thus, the digital signature function includes the authentication function.

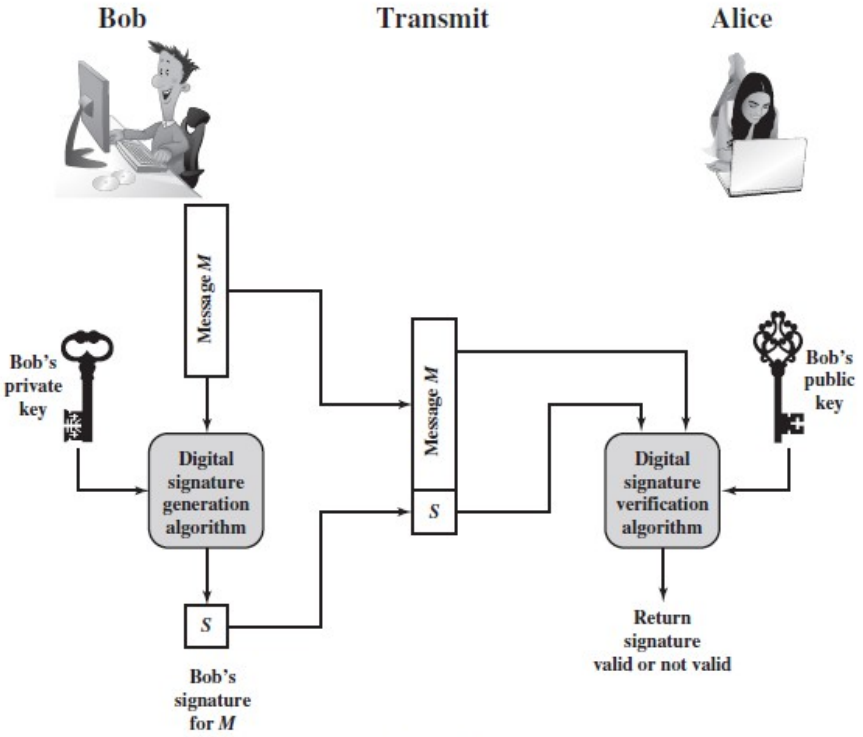


Figure 13.1 Generic Model of Digital Signature Process

Attacks and Forgeries

Here A denotes the user whose signature method is being attacked, and C denotes the attacker.

• ***Key-only attack:*** C only knows A's public key.

• ***Known message attack***: C is given access to a set of messages and their signatures.

• ***Generic chosen message attack***: C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic, because it does not depend on A's public key; the same attack is used against
everyone.

• ***Directed chosen message attack***: Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.

• ***Adaptive chosen message attack***: C is allowed to use A as an "oracle." This means that C may request from A signatures of messages that depend on previously obtained message-signature pairs.

Breaking a signature scheme is an outcome in which C can do any of the following with a non-negligible probability:

• ***Total break***: C determines A's private key.

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

Digital Signature Requirements

The following are the requirements for a digital signature.
• The signature must be a bit pattern that depends on the message being signed.
• The signature must use some information unique to the sender to prevent both forgery and denial.
• It must be relatively easy to produce the digital signature.
• It must be relatively easy to recognize and verify the digital signature.
• It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
• It must be practical to retain a copy of the digital signature in storage.

Elgamal Digital Signature Scheme

The Elgamal signature scheme involves the use of the private key for encryption and the public key for decryption
The global elements of **Elgamal digital signature** are a prime number q and a, which is a primitive root of q.

User A generates a private/public key pair as follows.

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = a^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, a, Y_A\}$.

To sign a message *M*,
User A first computes the hash $m = H(M)$, such that m is an integer in the range $0 \leq m \leq q - 1$. A then forms a digital signature as follows.

1. Choose a random integer K such that $1 \leq K \leq q - 1$ and $\gcd(K, q - 1) = 1$.That is, K is relatively prime to $q - 1$.
2. Compute $S1 = \alpha^K \bmod q$.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S2 = K^{-1}(m - X_A S1) \bmod (q - 1)$.
5. The signature consists of the pair $(S1, S2)$.

Any user B can verify the signature as follows.

1. Compute $V1 = \alpha_m \bmod q$.
2. Compute $V2 = (Y_A)^{S1} (S1)^{S2} \bmod q$.

The signature is valid if $V1 = V2$.

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, We choose a = 10.
Alice generates a key pair as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{XA} \bmod q = \alpha^{16} \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is {q, a, YA} = {19, 10, 4}.Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S1 = \alpha^{KA} \bmod q = 10^5 \bmod 19 = 3$
3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
4. $S2 = K^{-1} (m - X_A S1) \bmod (q - 1) = 11 (14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Bob can verify the signature as follows.

1. $V1 = \alpha^{m} \bmod q = 10^{14} \bmod 19 = 16$.
2. $V2 = (Y_A)^{S1}(S1)^{S2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$. Thus, the signature is valid.

Schnorr Digital Signature Scheme

The Schnorr signature scheme is based on discrete logarithms.
The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.
The main work for signature generation does not depend on the message and can be done during the idle time of the processor.
The message-dependent part of the signature generation requires multiplying a 2n-bit integer with an n-bit integer.
The scheme is based on using a prime modulus p, with $p - 1$ having a prime factor q of appropriate size; that is, $p - 1 = K \pmod q$.
Thus, p is a 1024-bit number, and q is a 160-bit number, which is also the length of the SHA-1 hash value.

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $\alpha^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

A user with private key s and public key v generates a signature as follows.

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$$e = \mathrm{H}(M \| x)$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Any other user can verify the signature as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = \mathrm{H}(M \| x')$.

To see that the verification works, observe that

$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$$

Hence, $\mathrm{H}(M \| x') = \mathrm{H}(M \| x)$.

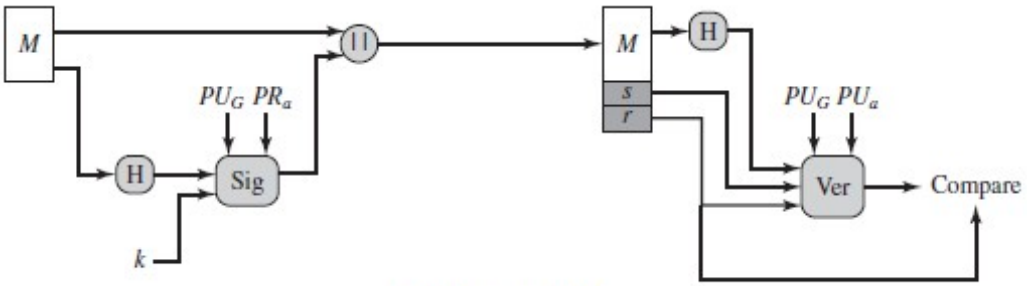
DSS

- Digital Signature Standard (DSS) is the digital signature algorithm (DSA) developed by the U.S. National Security Agency (NSA) to generate a digital signature for the authentication of electronic documents.
- DSA is a pair of large numbers that are computed according to the specified algorithm within parameters that enable the authentication of the signatory, and as a consequence, the integrity of the data attached.
- Digital signatures are generated through DSA, as well as verified. Signatures are generated in conjunction with the use of a private key; verification takes place in reference to a corresponding public key.
- Each signatory has their own paired public (assumed to be known to the general public) and private (known only to the user) keys. Because a signature can only be generated by an authorized person using their private key, the corresponding public key can be used by anyone to verify the signature.

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

DSA approach to Digital signature

- The hash code is provided as input to a signature function along with a random number k generated for this particular signature.
- The signature function also depends on the sender's private key (*PRa*) and a set of parameters known to a group of communicating principals.We can consider this set to constitute a global public key (*PUG*).1
- The result is a signature consisting of two components, labeled *s* and *r*.
- At the receiving end, the hash code of the incoming message is generated. Thisplus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (*PUa*), which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component *r* if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.



(b) DSA approach

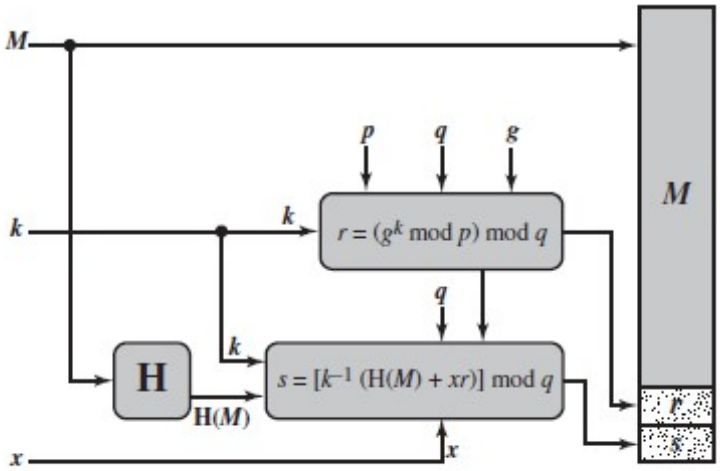
The Digital Signature Algorithm

- Based on the difficulty of computing discrete logarithms
- Three parameters that are public and can be common to a group of users
  - a.i. An *N*-bit prime number *q*
  - a.ii. A prime number with a length between 512 and 1024 bits such that *q* divides (*p* - 1).
  - a.iii. Finally, *g* is chosen to be of the form $h^{(p-1)/q}$ mod *p*, where *h* is an integer between 1 and (*p* - 1) with the restriction that *g* must be greater than 1.
- Each user selects a private key and generates a public key.

- The private key x must be a number from 1 to (q - 1) and should be chosen randomly or pseudo randomly.
- The public key is calculated from the private key as $y = g^x \bmod p$.
- Given the public key y, it is believed to be computationally infeasible to determine x, which is the discrete logarithm of y to the base g, mod p.
- **To create a signature**

  Calculate two quantities, r and s, that are functions of the public key components (p, q, g), the user's private key (x), the hash code of the message H(M), and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing.
- **To verify a signature**

  The receiver generates a quantity v that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the r component of the signature, then the signature is validated.
- Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover k from r or to recover x from s.
- A user could precalculate a number of values of r to be used to sign documents as needed.
- The only other somewhat demanding task is the determination of a multiplicative inverse, k^{-1}. Again, a number of these values can be precalculated.



(a) Signing

-

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$ i.e., bit length of N bits

g $= h(p-1)/q \bmod p$, where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature $= (r, s)$

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

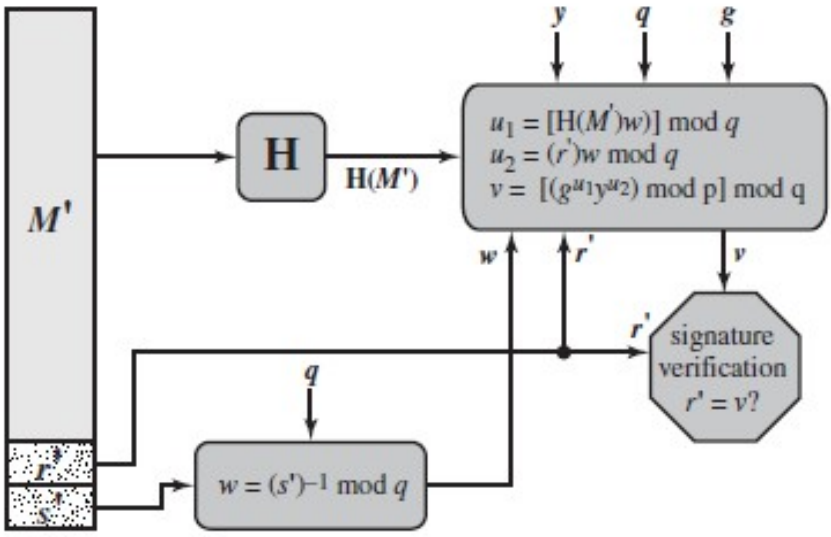
TEST: $v = r'$

M = message to be signed

H(M) = hash of M using SHA-1

M', r', s' = received versions of M, r, s

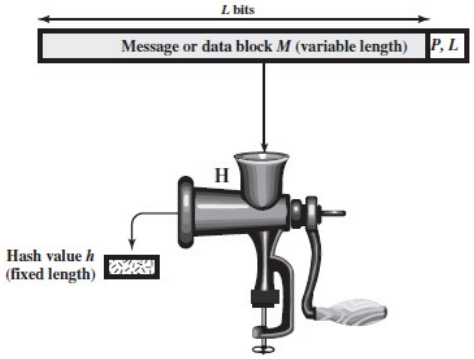
Figure 13.4 The Digital Signature Algorithm (DSA)



(b) Verifying

HASH FUNCTIONS

- A **hash function** H accepts a variable-length block of data *M* as input and produces a fixed-size hash value *h* = H(*M*).
- A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.
- The principal object of a hash function is data integrity.
- A change to any bit or bits in *M* results, with high probability, in a change to the hash code.
- The kind of hash function needed for security applications is referred to as a **cryptographic hash function**.
- A cryptographic hash function is an algorithm for which it is computationally infeasible to find either
    1. a data object that maps to a pre-specified hash result (the one-way property) or
    2. two data objects that map to the same hash result (the collision-free property).



P, L = padding plus length field

Figure 11.1 Cryptographic Hash Function; $h = H(M)$

Figure 11.1 depicts the general operation of a cryptographic hash function.

- Typically, the input is padded out to an integer multiple of some fixed length (e.g., 1024 bits), and the padding includes the value of the length of the original message in bits.
- The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.
- All hash functions operate using the following general principles.
    1. The input (message, file,etc.) is viewed as a sequence of *n*-bit blocks.
    2. The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function.
    3. One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as

$$C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

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

Security Requirements for Cryptographic Hash Functions

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

Table 11.1 Requirements for a Cryptographic Hash Function H

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

- The first three properties are requirements for the practical application of a hash function.
- The fourth property, **preimage resistant**, is the one-way property. It is easy to generate a code given a message, but virtually impossible to generate a message given a code.
- The fifth property, **second preimage resistant**, guarantees that it is impossible to find an alternative message with the same hash value as a given message. This prevents forgery when an encrypted hash code is used.
- A hash function that satisfies the first five properties is referred to as a weak hash function.
- If the sixth property, **collision resistant**, is also satisfied,then it is referred to as a strong hash function.
- A strong hash function protects against an attack in which one party generates a message for another party to sign.

- A function that is collision resistant is also second preimage resistant, but the reverse is not necessarily true.
- A function can be collision resistant but not preimage resistant and vice versa.
- A function can be preimage resistant but not second preimage resistant and vice versa.

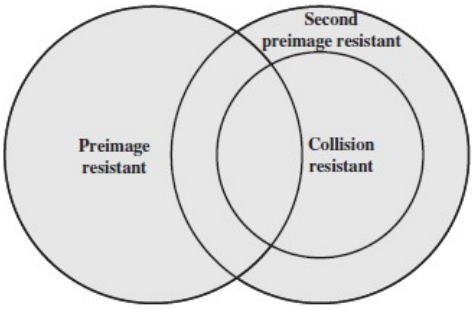


Figure 11.6 Relationship Among Hash Function Properties

- Cryptographic hash functions are commonly used for key derivation and pseudorandom number generation, and that in message integrity applications, the three resistant properties depend on the output of the hash function appearing to be random.
- Thus, it makes sense to verify that in fact a given hash function produces pseudorandom output.

Table 11.2 Hash Function Resistance Properties Required for Various Data Integrity Applications

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

-

ATTACKS ON HASH FUNCTIONS

- There are two categories of attacks on hash functions brute-force attacks and cryptanalysis.
- Brute-force attack depends only on the bit length of the hash value.
- A cryptanalysis, based on weaknesses in a particular cryptographic algorithm.

Preimage and Second Preimage Attacks

- For a preimage or second preimage attack,an adversary wishes to find a value *y* such that H(*y*) is equal to a given hash value *h*.
- The brute-force method is to pick values of *y* at random and try each value until a collision occurs.
- For an *m*-bit hash value, the level of effort is proportional to 2^m.
- The adversary would have to try, on average, 2^{m-1} values of y to find one that generates a given hash value *h*.

Collision Resistant Attacks

- An adversary wishes to find two messages or data blocks, *x* and *y*, that yield the same hash function:
- H(*x*) = H(*y*).
- Require considerably less effort than a preimage or second preimage attack. The effort required is explained by a mathematical result referred to as the **birthday paradox**.

Birthday paradox.

- If we choose random variables from a uniform distribution in the range 0 through $N - 1$, then the probability that a Repeated element is encountered exceeds 0.5 after $\sqrt{N}$ choices have been made.
- Thus, for an *m*-bit hash value, if we pick data blocks at random, we can expect to find two data blocks with the same hash value within $\sqrt{2^m} = \sqrt{2^{m/2}}$ attempts.
- The following strategy to exploit the birthday paradox in a collision resistant attack

1. The source, A, is prepared to sign a legitimate message x by appending the appropriate m-bit hash code and encrypting that hash code with A's private key (Figure 11.4a).
2. The opponent generates $2^{m/2}$ variations x' of x, all of which convey essentially the same meaning, and stores the messages and their hash values.
3. The opponent prepares a fraudulent message y for which A's signature is desired.
4. The opponent generates minor variations y' of y, all of which convey essentially the same meaning. For each y', the opponent computes $H(y')$, checks for matches with any of the $H(x')$ values, and continues until a match is found. That is, the process continues until a y' is generated with a hash value equal to the hash value of one of the x' values.
5. 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.

- The generation of many variations that convey the same meaning is not difficult.
- For example, the opponent could insert a number of “space-space-backspace” character pairs between words throughout the document.
- Variations could then be generated by substituting “space-backspace-space” in selected instances. Alternatively, the opponent could simply reword the message but retain the meaning.

To summarize, for a hash code of length m, the level of effort required, as we have seen, is proportional to the following.

Preimage resistant	2^m
Second preimage resistant	2^m
Collision resistant	$2^{m/2}$

As { the / — } Dean of Blakewell College, I have { had the pleasure of knowing / known } Cherise Rosetti for the { last / past } four years. She { has been / was } { a tremendous / an outstanding } { asset to / role model in } { our / the } school. I { would like to take this opportunity to / wholeheartedly } recommend Cherise for your { school's / — } graduate program. I { am / feel } { confident / certain } { that / — } { she / Cherise } will { continue to / — } succeed in her studies. { She / Cherise } is a dedicated student and

Cryptanalysis

- The way to measure the resistance of a hash algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.
- An ideal hash algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

iterated hash function,

- The structure of most hash functions
- 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.
- The hash function can be summarized as

$$\begin{aligned} CV_0 &= IV = \text{initial } n\text{-bit value} \\ CV_i &= \mathrm{f}(CV_{i-1}, Y_{i-1}) \quad 1 \le i \le L \\ \mathrm{H}(M) &= CV_L \end{aligned}$$

where the input to the hash function is a message M consisting of the blocks $Y_0, Y_1, \ldots, Y_{L-1}$.

- If the compression function is collision resistant, then so is the resultant iterated hash function.
- The structure can be used to produce a secure hash function to operate on a message of any length.
- The problem of designing a secure hash function reduces to that of designing a collision-resistant compression function that operates on inputs of some fixed size.

MD5 Message-digest Algorithm

- The MD5 message-digest algorithm was developed by Ronald Rivest at MIT in 1992.
- This algorithm takes a input message of arbitrary length and produces a 128-bit hash value of the message.
- The input message is processed in 512-bit blocks which can be divided into 16 32-bit subblocks.
- The message digest is a set of four 32-bit blocks, which concatenate to form a single 128-bit hash code.

The following steps are carried out to compute the message digest of the input message.

1 Append Padding Bits

- The message is padded so that its length (in bits) is congruent to 448 modulo 512.
- This padding is formed by appending a single ‘ 1’ bit to the end of the message, and then ‘ 0’ bits are appended as needed such that the length (in bits) of the padded message becomes congruent to 448 *(*= 512 − 64*)*, modulo 512.

2 Append Length

- A 64-bit representation of the original message length is appended to the result of the previous step.

- If the original length is greater than 2^{64}, then only the low-order 64 bits of the length are used for appending two 32-bit words.
- The length of the resulting message is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words.
- Let M[0 . . .$N - 1$] denote the word of the resulting message, with N an integer multiple of 16.

3 Initialise MD Buffer

- A four-word buffer represents four 32-bit registers (A, B, C and D).
- This 128-bit buffer is used to compute the message digest. These registers are initialised to the following values in hexadecimal (low-order bytes first):

A = 01 23 45 67
B = 89 ab cd ef
C = fe dc ba 98
D = 76 54 32 10

- These four variables are then copied into different variables: A as AA, B as BB, C as CC and D as DD.

4 Define Four Auxiliary Functions (F, G, H, I)

- F, G, H and I are four basic MD5 functions.
- Each of these four nonlinear functions takes three 32-bit words as input and produces one 32-bit word as output. They are, one for each round,

$$F(X, Y, Z) = (X \cdot Y) + (\overline{X} \cdot Z)$$
$$G(X, Y, Z) = (X \cdot Z) + (Y \cdot \overline{Z})$$
$$H(X, Y, Z) = X \oplus Y \oplus Z$$
$$I(X, Y, Z) = Y \oplus (X + \overline{Z})$$

Table 4.5 Truth table of four nonlinear functions

XYZ	FGHI
000	0001
001	1010
010	0110
011	1001
100	0011
101	0101
110	1100
111	1110

5 FF, GG, HH and II Transformations for Rounds 1, 2, 3 and 4

If M[k], $0 \leq k \leq 15$, denotes the kth sub-block of the message, and <<< s represents a left shift s bits, the four operations are defined as follows:

FF(a, b, c, d, M[k], s, i) : a = b + ((a + F(b, c, d) + M[k] + T[i] <<< s)

GG(a, b, c, d, M[k], s, i) : a = b + ((a + G(b, c, d) + M[k] + T[i] <<< s)

HH(a, b, c, d, M[k], s, i) : a = b + ((a + H(b, c, d) + M[k] + T[i] <<< s)

II(a, b, c, d, M[k], s, i) : a = b + ((a + I(b, c, d) + M[k] + T[i] <<< s)

Computation uses a 64-element table T[i], $i = 1, 2, \ldots, 64$, which is constructed from the sine function. T[i] denotes the ith element of the table, which is equal to the integer part of 4294967296 times abs(sin(i)), where i is in radians:

T[i] = integer part of $[2^{32} * |\sin(i)|]$

where $0 \leq |\sin(i)| \leq 1$ and $0 \leqslant 2^{32} * |\sin(i)| \leq 2^{32}$.

6 Computation of Four Rounds (64 Steps)

Each round consists of 16 operations. Each operation performs a nonlinear function on three of A, B, C and D.

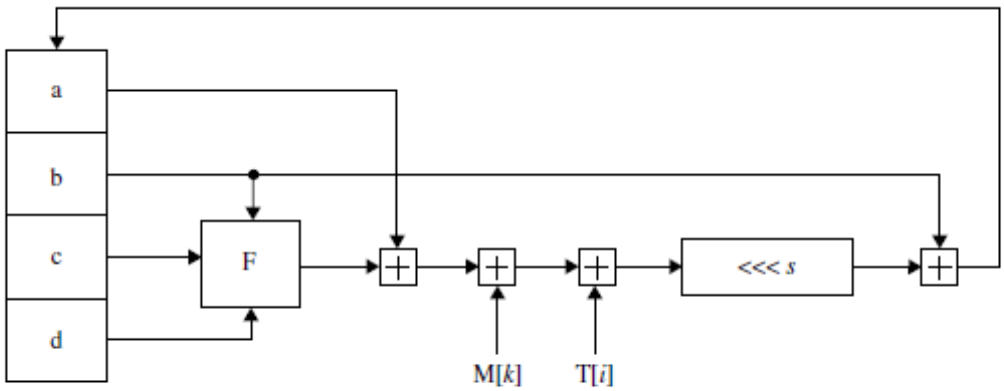


Figure 4.12 Basic MD5 operation.

Round 1

Let FF[a, b, c, d, M[k], s, i] denote the operation

$a = b + ((a + F(b, c, d) + M[k] + T[i]) <<< s)$.

Then the following 16 operations are computed:

FF[a, b, c, d, M[0], 7, 1], FF[d, a, b, c, M[1], 12, 2], FF[c, d, a, b, M[2], 17, 3],
FF[b, c, d, a, M[3], 22, 4], FF[a, b, c, d, M[4], 7, 5], FF[d, a, b, c, M[5], 12, 6],

FF[c, d, a, b, M[6], 17, 7], FF[b, c, d, a, M[7], 22, 8], FF[a, b, c, d, M[8], 7, 9],
FF[d, a, b, c, M[9], 12, 10], FF[c, d, a, b, M[10], 17, 11], FF[b, c, d, a, M[11], 22, 12],
FF[a, b, c, d, M[12], 7, 13], FF[d, a, b, c, M[13], 12, 14], FF[c, d, a, b, M[14], 17, 15],
FF[b, c, d, a, M[15], 22, 16]

Round 2

Let GG[a, b, c, d, $M[k]$, s, i] denote the operation

$a = b + ((a + G(b, c, d) + M[k] + T[i]) <<< s)$.

Then the following 16 operations are computed:

GG[a, b, c, d, M[1], 5, 17], GG[d, a, b, c, M[6], 9, 18], GG[c, d, a, b, M[11], 14, 19],
GG[b, c, d, a, M[0], 20, 20], GG[a, b, c, d, M[5], 5, 21], GG[d, a, b, c, M[10], 9, 22],
GG[c, d, a, b, M[15], 14, 23], GG[b, c, d, a, M[4], 20, 24], GG[a, b, c, d, M[9], 5, 25],
GG[d, a, b, c, M[14], 9, 26], GG[c, d, a, b, M[3], 14, 27], GG[b, c, d, a, M[8], 20, 28],
GG[a, b, c, d, M[13], 5, 29], GG[d, a, b, c, M[2], 9, 30], GG[c, d, a, b, M[7], 14, 31],
GG[b, c, d, a, M[12], 20, 32],

Round 3

Let HH[a, b, c, d, M[k], s, i] denote the operation

$a = b + ((a + H(b, c, d) + M[k] + T[i]) <<< s)$.

Then the following 16 operations are computed:

HH[a, b, c, d, M[5], 4, 33], HH[d, a, b, c, M[8], 11, 34], HH[c, d, a, b, M[11], 16, 35],
HH[b, c, d, a, M[14], 23, 36], HH[a, b, c, d, M[1], 4, 37], HH[d, a, b, c, M[4], 11, 38],

HH[c, d, a, b, M[7], 16, 39], HH[b, c, d, a, M[10], 23, 40], HH[a, b, c, d, M[13], 4, 41], HH[d, a, b, c, M[0], 11, 42], HH[c, d, a, b, M[3], 16, 43], HH[b, c, d, a, M[6], 23, 44], HH[a, b, c, d, M[9], 4, 45], HH[d, a, b, c, M[12], 11, 46], HH[c, d, a, b, M[15], 16, 47], HH[b, c, d, a, M[2], 23, 48],

Round 4

Let II[a, b, c, d, M[k], s, i] denote the operation

$a = b + ((a + I(b, c, d) + M[k] + T[i]) <<< s)$.

Then the following 16 operations are computed:

II[a, b, c, d, M[0], 6, 49], II[d, a, b, c, M[7], 10, 50], II[c, d, a, b, M[14], 15, 51], II[b, c, d, a, M[5], 21, 52], II[a, b, c, d, M[12], 6, 53], II[d, a, b, c, M[3], 10, 54], II[c, d, a, b, M[10], 15, 55], II[b, c, d, a, M[1], 21, 56], II[a, b, c, d, M[8], 6, 57], II[d, a, b, c, M[15], 10, 58], II[c, d, a, b, M[6], 15, 59], II[b, c, d, a, M[13], 21, 60], II[a, b, c, d, M[4], 6, 61], II[d, a, b, c, M[11], 10, 62], II[c, d, a, b, M[2], 15, 63], II[b, c, d, a, M[9], 21, 64],

After all of the above steps, A, B, C and D are added to their respective increments AA, BB, CC and DD, as follows:

$A = A + AA, B = B + BB$

$C = C + CC, D = D + DD$

and the algorithm continues with the resulting block of data. The final output is the concatenation of A, B, C and D.

SHA –I

- The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) for use with the Digital Signature Algorithm (DSA).
- The Secure Hash Standard (SHS) specifies a SHA-1 for computing the hash value of a message or a data file.
- When a message of any length of less than 2^{64} bits is input, the SHA-1 produces a 160-bit output called a message digest (or a hash code).
- The SHA-1 is secure because it is computationally impossible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.
- Any change to a message in transit will result in a different message digest, and the signature will fail to verify.
-

1 Message Padding

- The message padding is provided to make a final padded message a multiple of 512 bits.

- The SHA-1 sequentially processes blocks of 512 bits when computing the hash value (or message digest) of a message or data file that is provided as input. Padding is exactly the same as in MD5.
- First append a '1' followed by as many '0's as necessary to make it 64 bits short of a multiple of 512 bits, and finally a 64-bit integer is appended to the end of the zero appended message to produce a final padded message of length n × 512 bits.
- The 64-bit integer 'I' represents the length of the original message. Now, the padded message is then processed by the SHA-1 as n × 512 bit blocks.

2 Initialise 160-bit Buffer

The 160-bit buffer consists of five 32-bit registers (A, B, C, D and E). Before processing any blocks, these registers are initialised to the following hexadecimal values:

$H0$ = 67 45 23 01
$H1$ = ef cd ab 89
$H2$ = 98 ba dc fe
$H3$ = 10 32 54 76
$H4$ = c3 d2 e1 f0

The first four values are the same as those used in MD5. The only difference is the use of a different rule for expressing the values, i.e. high-order octets first for SHA and low-order octets first for MD5.

4.4.3 Functions Used

A sequence of logical functions $f_0, f_1, \ldots, f_{79}$ is used in SHA-1. Each function $f_t, 0 \leqslant t \leqslant 79$, operates on three 32-bit words B, C and D, and produces a 32-bit word as output. Each operation performs a nonlinear operation of three of A, B, C and D, and then does shifting and adding as in MD5. The set of SHA primitive functions, f_t (B, C, D) is defined as follows:

$$f_t(B, C, D) = (B \cdot C) + (\overline{B} \cdot D), 0 \leqslant t \leqslant 19$$

$$f_t(B, C, D) = B \oplus C \oplus D, 20 \leqslant t \leqslant 39$$

$$f_t(B, C, D) = (B \cdot C) + (B \cdot D) + (C \cdot D), 40 \leqslant t \leqslant 59$$

$$f_t(B, C, D) = B \oplus C \oplus D, 60 \leqslant t \leqslant 79$$

where $B \cdot C$ = bitwise logical 'AND' of B and C
$B \oplus C$ = bitwise logical XOR of B and C
$\overline{B}$ = bitwise logical 'complement' of B
$\boxplus$ = addition modulo 2^{32}

As you can see, only three different functions are used. For $0 \leq t \leq 19$, the function f_t acts as a conditional: if B then C else D. For $20 \leq t \leq 39$ and $60 \leq t \leq 79$, the function f_t is true if two or three of the arguments are true. Table 4.7 is a truth table of these functions.

4.4.4 Constants Used

Four distinct constants are used in SHA-1. In hexadecimal, these values are given by

$$K_t = 5a827999, \quad 0 \leq t \leq 19$$

$$K_t = 6ed9eba1, \quad 20 \leq t \leq 39$$

$$K_t = 8fbbcdc, \quad 40 \leq t \leq 59$$

$$K_t = ca62c1d6, \quad 60 \leq t \leq 79$$

4.4.5 Computing the Message Digest

The message digest is computed using the final padded message. To generate the message digest, the 16-word blocks (M_0 to M_{15}) are processed in order. The processing of each M_i involves 80 steps. That is, the message block is transformed from 16 32-bit words (M_0 to M_{15}) to 80 32-bit words (W_0 to W_{79}) using the following algorithm.

Divide M_i into 16 words W_0, W_1, ..., W_{15}, where W_0 is the leftmost word. For $t = 0$ to 15, $W_t = M_t$. For $t = 16$ to 79, $W_t = \mathrm{S}^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$.

Let $\mathrm{A} = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$. For $t = 0$ to 79 do

$$\mathrm{TEMP} = \mathrm{S}^5(\mathrm{A}) + \mathrm{F}_t(\mathrm{B}, \mathrm{C}, \mathrm{D}) + \mathrm{E} + W_t + K_t;$$

$$\mathrm{E} = \mathrm{D}; \mathrm{D} = \mathrm{C}; \mathrm{C} = \mathrm{S}^{30}(\mathrm{B}); \mathrm{B} = \mathrm{A}; \mathrm{A} = \mathrm{TEMP}$$

where:

A, B, C, D, E: Five words of the buffer
t: Round number, $0 \leq t \leq 79$
S^i: Circular left shift by i bits
W_t: A 32-bit word derived from the current 512-bit input block
K_t: An additive constant
$\boxplus$: Addition modulo 2^{32}

After all N 512-bit blocks have been processed, the output from the Nth stage is the 160-bit message digest, represented by the five words H_0, H_1, H_2, H_3 and H_4.

The SHA-1 operation looking at the logic in each of 80 rounds of one 512-bit block is shown in Figure 4.13.

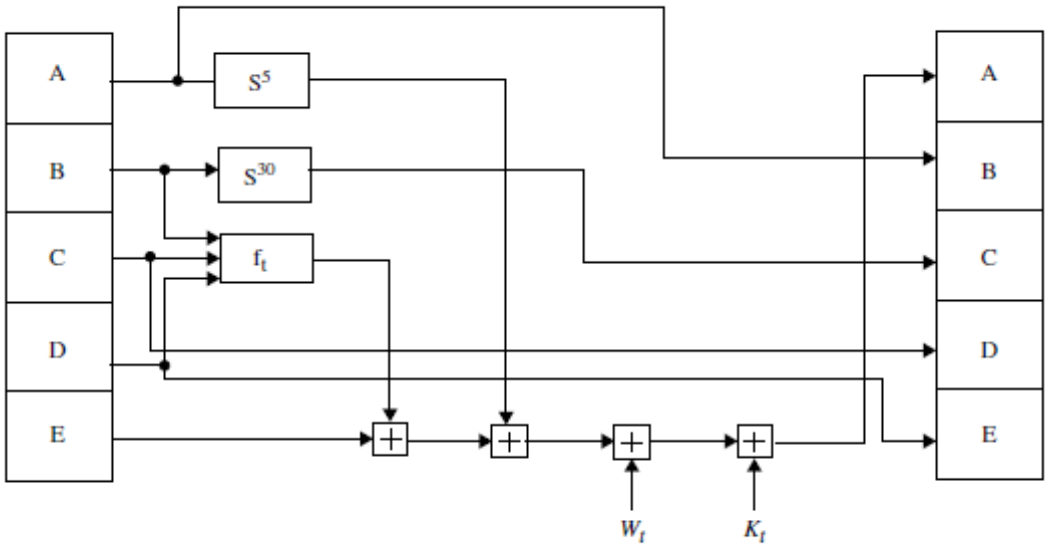


Figure 4.13 SHA-1 operation.

After all 512-bit blocks have been processed, the output represented by the five words, H_0, H_1, H_2, H_3 and H_4 is the 160-bit message digest as shown below:

H_0: 48878397

H_1: 9801d679

H_2: 394bd834

H_3: 28c28e41

H_4: 2b8dee05

The 160-bit message digest is then the data concatenation of $\{H_i\}$:

$H_0 \| H_1 \| H_2 \| H_3 \| H_4 = 488783979801d679394bd83428c28e412b8dee05$

- The digitised document or message of any length can create a 160-bit message digest which is produced using the SHA-1 algorithm.
- Any change to a digitised message in transit results in a different message digest. In fact, changing a single bit of the data modifies at least half of the resulting digest bits.
- Furthermore, it is computationally impossible to find two meaningful messages that have the same 160-bit digest. On the other hand, given a 160-bit message digest, it is also impossible to find a meaningful message with that digest.

The main structural differences between **SHA 1 AND MD5** are the following:

- SHA-1 has a larger state: 160 bits vs 128 bits.
- SHA-1 has more rounds: 80 vs 64.
- SHA-1 rounds have an extra bit rotation and the mixing of state words is slightly different (mostly to account for the fifth word).
- Bitwise combination functions and round constants are different.
- Bit rotation counts in SHA-1 are the same for all rounds, while in MD5 each round has its own rotation count.
- The message words are pre-processed in SHA-0 and SHA-1. In MD5, each round uses one of the 16 message words "as is"; in SHA-0, the 16 message words are expanded into 80 derived words with a sort of word-wise linear feedback shift register. SHA-1 furthermore adds a bit rotation to this word derivation.

A. *Differences between MD5 and SHA Algorithms.*

Table 1:- Comparison between MD5 and SHA

Keys For Comparison	MD5	SHA
Security	Less Secure than SHA	High Secure than MD5
Message Digest Length	128 Bits	160 Bits
Attacks required to find out original Message	2^{128} bit operations required to break	2^{160} bit operations required to break
Attacks to try and find two messages producing the same MD	2^{64} bit operations required to break	2^{80} bit operations required to break
Speed	Faster, only 64 iterations	Slower than MD5, Required 80 iterations
Successful attacks so far	Attacks reported to some extents	No such attach report yet

B. *Similarities between MD5 and SHA Algorithms.*

Table 2:-Similarities between MD5 and SHA

Keys For Similarities	MD5	SHA
Padding	✓	✓
Message bit	✓	✓
Members (Hash Family)	✓	✓
Resource Utilization (same)	✓	✓
Fingerprint	✓	✓

UNIT IV

KEY MANAGEMENT AND DISTRIBUTION

Authentication applications:

- Developed to support application-level authentication and digital signatures
- Most widely used services:
  - Kerberos
  - X.509
- Kerberos – a private-key authentication service
- X.509 – a public-key directory authentication service

Kerberos:

- **Kerberos** is an authentication protocol. In Greek mythology Kerberos is a multiheaded dog which keeps intruders away. Here head of Kerberos represents three components used to gurad gate of network these are - authentication accounting and audit.
- Kerberos allows centralized authentication schemes amongs the users and sever. Kerberos does not employ public - key encryption but uses symmetric encryption. Commonly used versions of Kerberos are version 4 and version 5. Kerberos provides third party authentication service.
- Kerberos is a mature network authentication protocol developed at MIT. Kerberos allows entities to communicate securely over networks without having to transmit passwords in clear text.
- Kerberos is an authentication service designed for use in distributed environment. It makes use of a trusted third part authentication service that enables clients and servers to establish authenticated communication.
- Kerberos provides a centralized authentication server whose function is to authenticate user to servers and also servers to user.
- Kerberos based upon symmetric encryption i.e. DES.

Requirements for Kerberos

1. Secure
2. Reliable
3. Transparent
4. Scalable

Kerberos has been designed to be a authentication server with the following characteristics

1. **Secure**: no eavesdroper should be able to gather enough information to impersonate a Kerberos user.
2. **Reliable**: Because Kerberos is needed for the overall availability of the system, it needs to be distributed.
3. **Transparent:** Using Kerberos should not be visible to the user beyond the entering of a password.
4. **Scalable:** Kerberos should be capable of supporting a large number of clients.

To support these goals, Kerberos is a trusted third party authentication server that implements a variant of Needham-Schroeder. Kerberos is typically used to allow users to access networked resources. This adds another requirement to the list:

5. Users should not have to enter a password everytime that a new service is required.

Kerberos Version 4

- Kerberos version 4 uses DES for providing authentication service. Some aspect of version 4 are

A) Simple Authentication Dialogue.

B) More Secure Authentication Dialogue.

1 Simple Authentication Dialogue

- For a secure transaction, server should confirm the client and its request. In unprotected network it creates burden on server, therefore an authentication server (AS) is used. The authentication server (AS) maintains password of all users in centralized database. Also the authentication server shares a unique secret key with each server.
- Let

  Client is represented as C

  Authentication server is represented as AS

  Server is represented as V

  Identifier of user on C is represented as ID_C

  Identifier of V is represented as ID_V

  Password of user on C is P_C

  Network address of C is represented as AD_C

  Secret encryption key shared by AS and V is K_V

Then consider a hypothetical dialogue.

	Sender and receiver			Contents of message
1.	C	AS	:	$ID_C \| \| P_C \| \| ID_V$
2.	AS	C	:	Ticket
3.	C	V	:	$ID_c \| \|$ Ticket
4.	Ticket		=	$E [K_V , (ID_C \| \| AD_C \| \| ID_V)]$

Explanation

1. **Client C logs on to workstation requesting to access to server V :** The workstation requests user's password and sends message to AS including user ID + server ID + user password. The AS checks this message with database and verifies it.
2. **AS issues ticket :** On verifying the tests. AS issues ticket containing user ID + server ID + network address.
3. **Client C applies server V :** With this ticket, client C asks server V for access. Server V decrypts the ticket and verify the authenticity of data then grants the requested service. In above hypothetical dialogue, symbol || represents concatenation.

2 Secure Authentication Dialogue

- Kerberos version 4 protocol ensures secure authentication dialogue involving three sessions.

[I] Authentication Service - Exchange to obtain ticket-granting ticket.

[II] Ticket-granting Service - Exchange to obtain service granting ticket.

[III] Client/server authentication - Exchange to obtain service.

- Each of the above session has two steps, as shown in table below.

Session	Step	Sender-Receiver
[I]	1. 2.	C AS AS C
[II]	3. 4.	C TGS (Ticket-granting server) TGS C
[III]	5.	C V V C

Overview of Kerberos:

- A user logs on to a workstation using a password that is also known by the Key Distribution Center (KDC). The KDC authenticates the user and issues a Ticket Granting Ticket (TGT) that the client can use to request service from an application server.
- When the user accesses a remote system in a Kerberos environment, the user's workstation obtains a service ticket using the previously acquired TGT and passes the service ticket to the remote application (such as **telnet** or **ftp**) for authentication. The remote application server then validates the service ticket using a secret key that is known only by the KDC and the application server.

1. The client requests for a TGT: The client requests login credentials from the Authentication Server (AS) including a Ticket Granting Ticket (TGT) to be used for accessing the Ticket Granting Service (TGS).
2. The client receives the TGT: The KDC sends login credentials to the client, which comprises a Ticket Granting Ticket (TGT) and a session key. The TGT contains a copy of the session key and data identifying the client. The TGT is encrypted with a secret key known only to the KDC, and the session key is encrypted with the client's secret key, derived from the user's password.
3. The client requests for a Service Ticket: When the user invokes an application, the client requests a Service Ticket for the application server from the Ticket Granting Service (TGS) using its previously granted TGT.

4. The client receives the service ticket: The ticket granting service (TGS) encloses a service ticket in a response to the client and encrypts the response using the session key. The service ticket itself is encrypted with the key that the KDC shares with the application server, and contains the client's identity, the session key, timestamp and other information.
5. The application client requests for service: The client extracts the service ticket using the session key, creates an authenticator record with the session key, and sends the service ticket and the authenticator record to the application server.
6. The application client authenticates the application server: The application server decrypts the service ticket, extracts the session key, and uses this key to verify the authenticator record. If mutual authentication is required, the application server creates an authenticator by encrypting the timestamp enclosed in the service ticket using the session key and sends the authenticator to the client. If the client can successfully decrypt this message from the server and determines that the timestamp matches the timestamp the client received from the KDC, the server is authenticated.

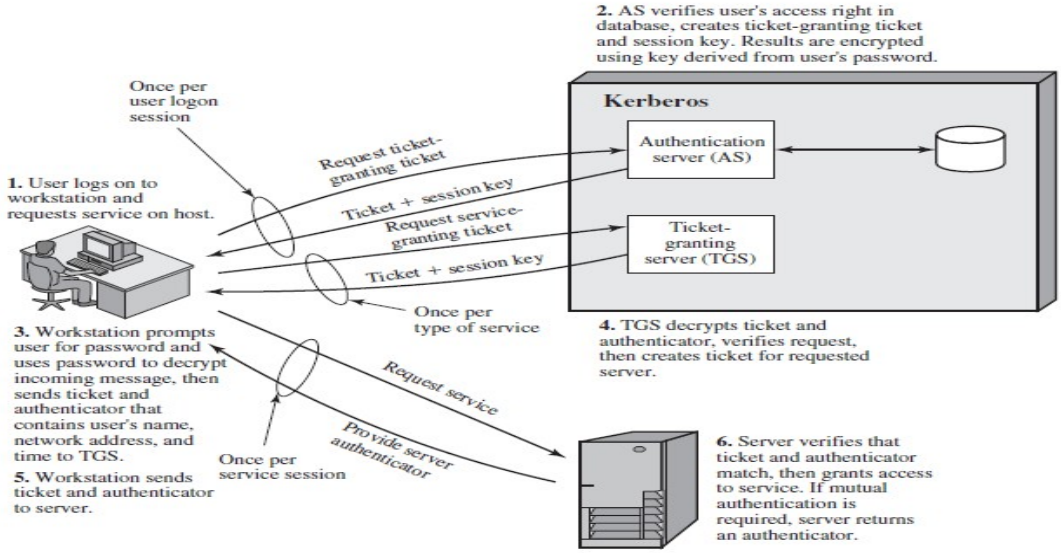


Figure 15.1 Overview of Kerberos

Kerberos 4 Authentication Dialogue

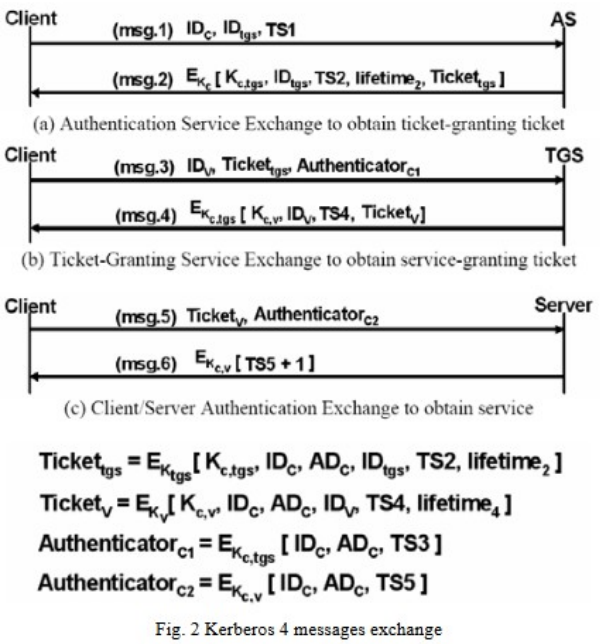


Fig. 2 Kerberos 4 messages exchange

Kerberos Version 4 messages exchange is shown in Fig. 2. Fig. 2 (a) shows the technique for distributing the session key. The client sends a message to the AS requesting access to the TGS. The AS responds with a message, encrypted with a key derived from the user's password (K

C) that contains the TGS ticket ([24] describes the password to key transformation technique that is presented by the standard specification). The encrypted message also contains a copy of the session key, Kc,tgs, where the subscripts indicate that this is a session key for C and TGS. Because this session key is inside the message encrypted with KC, only the client can read it. The same session key is included in the ticket, which can be read only by the TGS since it is encrypted by the TGS key Ktgs. Thus, the session key has been securely delivered to both the C and the TGS.

Here, we will focus on some messages' elements (the details can be found in [24]). The keys Kc,tgsand Kc,vare the session keys; where the subscripts indicate the communicating parties. Lifetime2 and lifetime4 are the lifetime of the TGS ticket and the server ticket respectively.

Finally, at the conclusion of this process, the client and server share a secret session key Kc,v

Kerberos 5 Authentication Dialogue

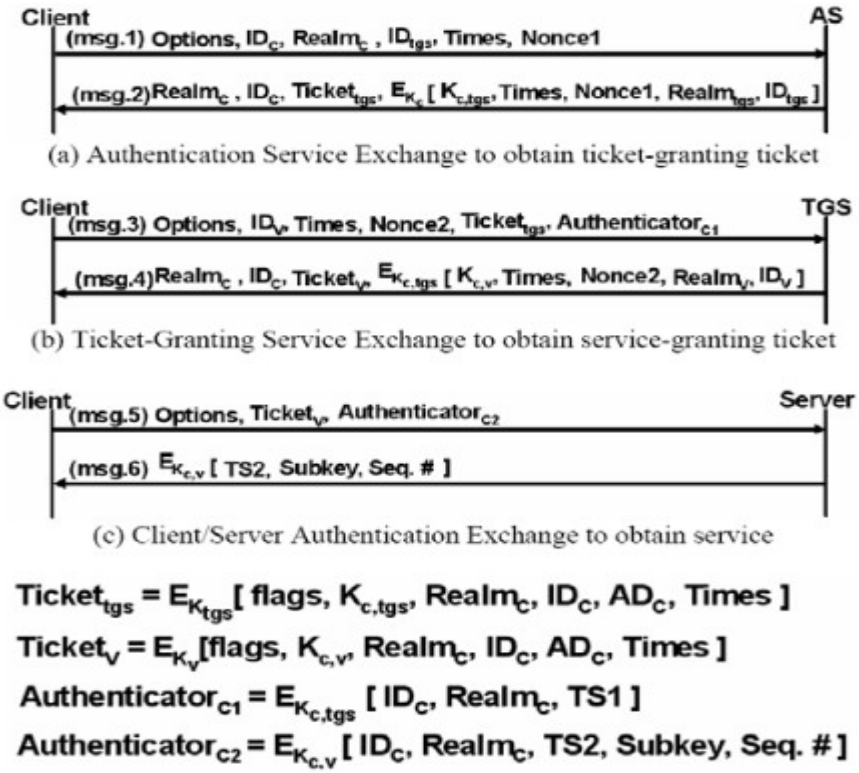


Fig. 3 Kerberos 5 messages exchange

Kerberos 5 messages exchange is shown in Fig. 3. This is best explained by comparison with version 4 (Fig. 2). In message (1), the following new elements are added:

•Realm: Indicates the realm of the client. Where the realm represents the nodes that are managed by a single KDC; i.e. share the same Kerberos database.

•Options: Used to request that certain flags be set in the returned ticket. These flags are an added feature in Kerberos 5.

•Times: Used by the client to request the following time settings in the ticket:

ofrom: the desired start time for the requested ticket.

otill: the requested expiration time for the requested ticket.

ortill: this field is only present in tickets that have the RENEWABLE flag set in the flags field. It indicates the maximum end-time that may be included in a renewal.

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

Let us **now compare the ticket-granting service exchange for versions 4 and 5**. We see that message (3) in Fig. 3 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1). 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. If the client selects a sub-session key, care must be taken to ensure the randomness of the selected key.

•Sequence number: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session (to detect replays). After that, the server responds with message (6). This message includes the timestamp from the authenticator. The subkey field, if present, overrides the subkey field of message (5). The optional sequence number field specifies the starting sequence number to be used by the client.

Realms

A Kerberos server needs entries for all users and all service providers in its database. We can call these entities the *realm* of Kerberos. However, users in one realm might want to access services in another Kerberos' realm. The solution is to have the TGS of the second Kerberos be a registered service provider (providing service tickets) for the first Kerberos. A client in the first realm can use the super ticket to access the TGS of the first Kerberos for a ticket to the TGS of the second Kerberos in order to obtain a ticket for the service provider. This approach works well with a small number of Kerberi (Kerberoi?) but does not scale well.

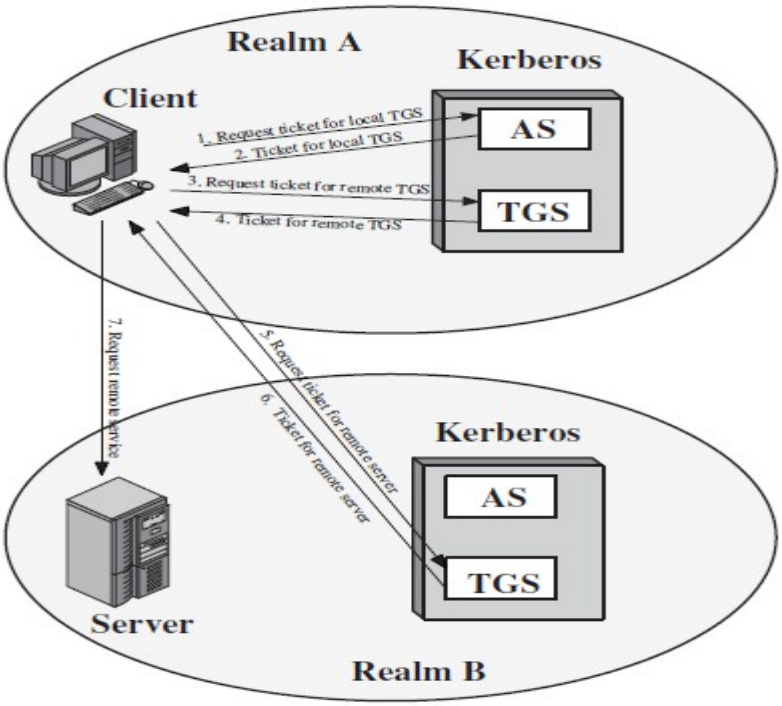


Figure 15.2 Request for Service in Another Realm

Differences between Versions 4 and 5 :

Version 5 is intended to address the limitations of version 4. Let us briefly discuss the ifferences between the two versions:

1.Encryption system dependence: Version 4 requires the use of DES. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used.

2.Internet protocol dependence: Version 4 requires the use of Internet Protocol (IPv4) addresses. In version 5, network address is tagged with type and length. This allows any network address type to be used.

3.Ticket lifetime: Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is 256 x 5 = 1280 minutes. In version 5, tickets include an explicit start and end times, allowing tickets with arbitrary lifetimes.

4.Authentication forwarding: Version 4 does not allow credentials issued to one client to be forwarded and used by some other clients. For example, a client issues a request to a print server that then accesses the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.

5.Double encryption: Note in Fig. 2 that tickets provided to clients in messages (2) and (4) are encrypted twice, once with the secret key of the target server and then again with a secret key

known to the client. The second encryption is not necessary and is computationally wasteful. It is avoided in version 5.

6.PCBC encryption: Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC) ([24] describes this mode of operation). Security problems have been demonstrated in that mode [11]. Version 5 makes use of the standard CBC mode for encryption.

7.Session keys: Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In addition, the session key may subsequently be used by the client and the server to protect messages passed during that session. However, because the same ticket may be used repeatedly to gain service from a particular server, there is the risk that an opponent will replay messages from an old session to the client or the server. In version 5, it is possible for a client and server to negotiate a sub-session key, which is to be used only for that one connection. A new access by the client would result in the use of a new sub-session key.

8.Password attacks: Both versions are vulnerable to a password guessing attack. The message from the AS to the client includes material encrypted with a key based on the client's password. An opponent can capture this message and attempt to decrypt it by trying various passwords. If the result of a test decryption is of the proper form, then the opponent has discovered the

client's password and may subsequently use it to gain authentication credentials from Kerberos. Remember that when a user requests the ticket-granting ticket, the answer is returned encrypted with KC, a key derived by a publicly-known algorithm from the user's password.

Kerberos Drawbacks

The protocol weaknesses can be summarized as follows:

1. Kerberos requires continuous availability of the KDC. When the Kerberos server is down, the system will be vulnerable to the single point of failure problem. This can be mitigated by using multiple Kerberos servers.

2.The system clocks of the hosts that are involved in the protocol should be synchronized. The tickets have a time availability period and if the host clock is not synchronized with the Kerberos server clock, the authentication will fail. In practice, Network Time Protocol daemons are usually used to keep the host clocks synchronized.

3."Password guessing" attacks are not solved by Kerberos. If a user chooses a poor password, it is possible for an attacker to successfully mount an offline dictionary attack by repeatedly attempting to decrypt messages obtained which are encrypted under a key derived from the user's password.

4.There are no standards for the administration of the Kerberos protocol. This will differ

Between server implementations.

X.509 Authentication Service

ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) recommendation X.509 is a part of the X.500 series of recommendations that define a directory service. The directory is a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users.

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. 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 variety of contexts (SSL, SET, etc.).

A third version of X.509 was issued in 1995 and revised in 2000.

Certificates

The heart of X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certificate authority (CA) and placed in the directory by the CA or by the user. The directory itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for

users to obtain certificates.Figure 14.3a shows the general format of a X.509 certificate.

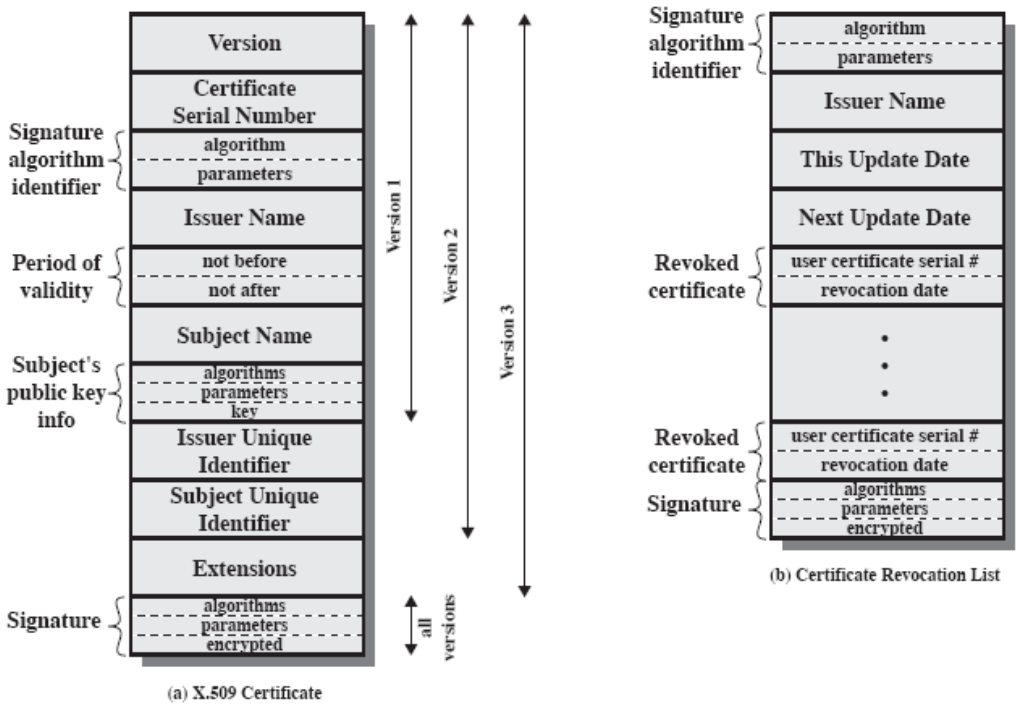


Figure 14.3 X.509 Formats

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

2 Serial number: An integer value, unique within the issuing CA, that is unambiguously associated with this certificate

3 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

4 Issuer name: X.500 name of the CA that created and signed this certificate (about X.500 names see, for example, http://java.sun.com/products/jndi/tutorial/ldap/models/x500.html)

5 Period of validity: Consists of two dates: the first and the last on which certificate is valid

6 Subject name: The name of the user to whom this certificate refers. That is, this

certificate certifies the public key of the subject who holds the corresponding private key

7 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

8 Issuer unique 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

9 Subject unique identifier: An optional bit string used to identify uniquely the subject in the event the X.500 name has been reused for different entities

10 Extensions: A set of one or more extension fields. Extensions were added in version 3 and are discussed later

11 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 identifier

The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used.

The standard uses the following notation to define a certificate:

$CA\langle\langle A\rangle\rangle = CA\{V,SN,AI,CA,T_A,A,Ap\}$

Where

$Y\langle\langle X\rangle\rangle$ = 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.

Obtaining a User's Certificate

User certificates generated by a CA have the following characteristics:

1 Any user with access to the public of the CA can verify the user public key that was encrypted

2 No party other than the CA can modify the certificate without this being detected

Because 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. In addition, a user can transmit his certificate directly to other users. In either case, once B is in possession of A's certificate, B has confidence that message it encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable.

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 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 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 are arranged in a hierarchy so that navigation is straightforward.

Figure 14.4, taken from X.509, is an example of such a hierarchy. The connected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates

maintained in the directory for each CA entry. The directory entry for each CA includes two types of certificates:

- Forward certificates: Certificates of X generated by other CAs
- Reverse certificates: Certificates generated by X that are the certificates of other CAs

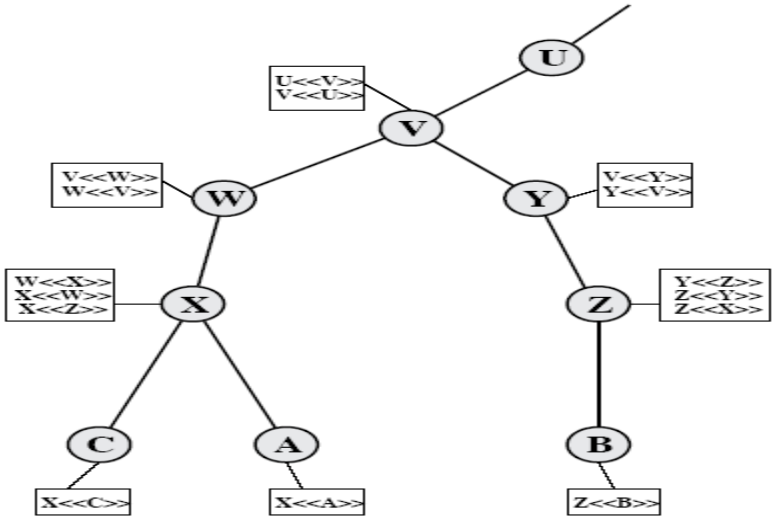


Figure 14.4 X.509 CA Hierarchy: a Hypothetical Example

In this example, user A can acquire the following certificates from the directory to establish a certification path to B:

X<<W>> W<<V>> V<<Y>> 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 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>>

B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.

Revocation of Certificates

Recall from Figure 14.3 that each certificate includes a period of validity. Typically, a new certificate is issued before the expiration of the old one. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:

1. The user's private key is assumed to be compromised
2. The user is no longer certified by this CA
3. The CA's certificate is assumed to be compromised

Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. These lists should be posted on the directory.

Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes (Figure 14.3b) the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate. Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.

When a user receives a certificate in a message, the user must determine whether the certificate has been revoked. The user could check the directory each time a certificate is received. To avoid the delays (and possible costs) associated with directory searches, it is likely that the user would maintain a local cache of certificates and lists of revoked certificates.

UNIT-V

E-mail Security: Security Services for E-mail

Privacy-only recipient reads the message.

Authentication-assuring the recipient identity to sender.

Integrity- assuring the recipient that message is not altered.

Non-repudiation-ability to prove to third party that sender has send the message.

Proof of submission-verification to sender that message given to mail delivery system

Proof of delivery-verification that recipient received message.

Message flow confidentiality-cannot determine whether a message flow was there

Anonymity-ability to send message so that recipient cant find sender identity.

Containment-ability of network to maintain security information from leaking.

Audit-ability of n/w to record events relavant to security.

Accounting-ability of mail system to maintain system usage statistics.

Self destruct-option for sender to specify when msg should be destroyed after delivery.

Message sequence integrity-assuring sequence of msg delivery.

Establishing keys

Establishing public keys:-

- Sender assumes receiver already has certificate or obtain certificate when necessary.
- Public key is known by
    - Known by out-of-band mechanism and install in workstation.
    - Obtain through PKI
    - Email system piggybacks certificates with message

Establishing secret keys:-

- Through private communication,I,e meeting in person ,talking over phone
- Obtain ticket from KDC and use it with message.

Email Privacy

Email Privacy – needed because:

Eavesdropper can easily listen especially at "No Such Agency"

Mail can be rerouted never to reach intended recipient

Conflicts employee wants privacy, company wants assurance employee is not giving away company secrets

End to end Privacy:

Problem: how to encrypt lots of copies to multiple recipients?

Secret key encryption is preferable since it is 1000 times faster

Not desirable to use a longterm more than necessary.Use expensive long term keys as little as possible.

Hence: sender may choose a secret S only for encrypting message msg encrypted with S + S encrypted with public key > recipient.S encrypted multiple times with recipients' public keys

To: prakash, masterblaster, cokane

From: franco

Key-info: prakash-7567484385785467

Key-info: masterblaster-734478868274684

Key-info: cokane-9062346667642424

Msg-info: jkdiuqwydfkjhjdfreuigfkjsdfkjsyfuieihfuigf

Privacy with Distribution List Exploders:

Problem: sender may not know public keys of recipients

Sender must have a key K shared with the distribution list exploder.

Sender encrypts message with a secret S and sends it with S encrypted using K to the list exploder.

Distribution list exploder decrypts S then reencrypts it with the keys of recipients (without decrypting the email?) and sends the email forward (possibly to other distribution list exploders).

But now sender loses some assurance that the message arrives as intended.

Authentication of the Source:

Prevent C from sending mail to B with 'From: A'

Public Keys:

Sender signs hash of message with its private key.

Works on multiple messages (same signature!).

Public key might be sent with the message with a chain of certificates.

Secret Keys:

Sender computes a MAC: one of

CBC residue of the message computed with shared secret

Hash of shared secret appended to message

Encrypted message digest of message

Multiple emails: use 3rd method compute MD once, then encrypt for each addressee.

Authentication of the Source Distribution Exploders:

Public keys: Just forward the messages as is, use sender's public key to authenticate

Secret keys: Sender cannot be assumed to share secrets with all recipients or know who all the recipients are Distribution list exploder must remove sender's authentication information from emails and replace it with its own Distribution list exploder must verify the source of the email because recipients cannot do that themselves although they can authenticate the exploder Exploder may need to include the name of the sender in the body of the encrypted email.

Message Integrity:

Source authentication methods also provide message integrity

Does it make sense to provide integrity without authentication?

Application: send a ransom note

Message integrity without source authentication meaningless without encryption since someone could replace the message with a completely different one and the recipient could not tell

Can't do message integrity check without source authentication with secret key technology since both parties must know each other to be able to use the same secret.

NonRepudiation:

With public keys easy to provide non repudiation authentication

With secret keys easy to provide repudiable authentication

Public Keys: plausible deniability [{S}receiver]sender, MAC,msg Sender chooses secret S and encrypts S with receiver's public key Sender signs result with its private key. Then Sender uses S to compute a MAC for the message

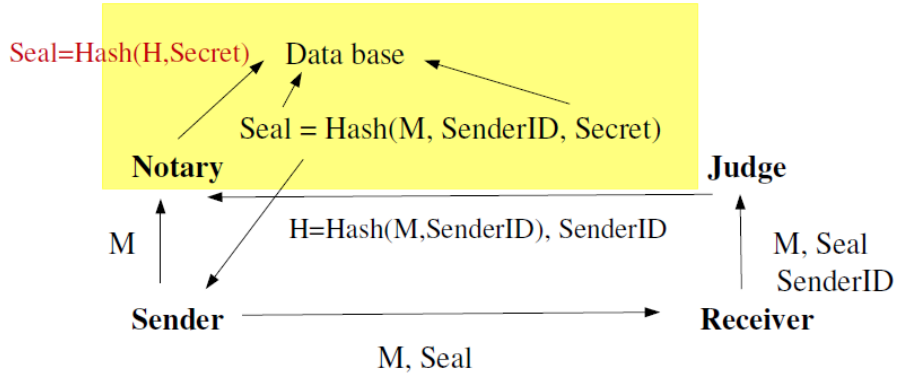
(e.g. CBC/DES residue)

Sender sends MAC, signed, encrypted S, message to recipient Recipient authenticates the sender via signature and computing the MAC of the message with S .

Can prove a message was received from sender using S but Can't prove the particular message was received because the Recipient can use [{S}receiver]sender to MAC any message.

Non-Repudiation:

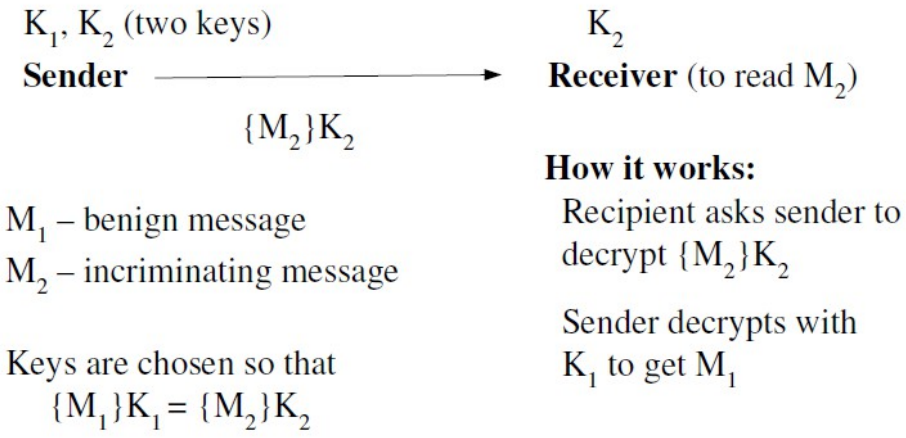
Secret Keys: non-repudiation



Proof: Receiver sends seal, M, sender ID to Judge
Judge takes Hash(M, SenderID), sends with SenderID to notary
Notary creates the seal and checks if it's in the database

Non-Repudiation:

Secret Keys: plausible deniability



PRETTY GOOD PRIVACY

PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

Zimmermann has done the following:

- Selected the best available cryptographic algorithms as building blocks.
- Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.
- Made the package and its documentation, including the source code, freely
- Entered into an agreement with a to provide a fully compatible, low-cost commercial version of PGP.

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

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

Notation

K_s = session key used in symmetric encryption scheme
P_{Ra} = private key of user A, used in public-key encryption scheme
P_{Ua} = public key of user A, used in public-key encryption scheme
EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation
Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format

Operational Description

consists of four services: authentication, confidentiality, compression, and e-mail compatibility

Authentication

- The sender creates a message.
- SHA-1 is used to generate a 160-bit hash code of the message.
- The hash code is encrypted with RSA using the sender's private key, and the result is pretended to the message.
- The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
- The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.
- The combination of SHA-1 and RSA provides an effective digital signature scheme.
- Signatures can be generated using DSS/SHA-1.
- Detached signatures are supported. A detached signature may be stored and transmitted separately from the message it signs.
- A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent virus infection.
- Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract.

Confidentiality

- Provided by encrypting messages to be transmitted or to be stored locally as files.
- The symmetric encryption algorithm CAST-128 may be used. Alternatively, IDEA or 3DES may be used. The 64-bit cipher feedback (CFB) mode is used.
- Each symmetric key is used only once. A new key is generated as a random 128-bit number for each message.

- The session key is bound to the message and transmitted with it. To protect the key, it is encrypted with the receiver's public key.
- PGP uses a variant of Diffie-Hellman that does provide encryption/decryption, known as ElGamal
- The sequence, for confidentiality can be described as follows.

1. The sender generates a message and a random 128-bit number session key for this message only.
2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
3. The session key is encrypted with RSA using the recipient's public key and is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.

- Several observations may be made
  1. To reduce encryption time, the combination of symmetric and public-key encryption is used.
  2. Use of the public-key algorithm solves the session-key distribution problem.
  3. given the store-andforward nature of electronic mail, the use of handshaking to assure that both sides have the same session key is not practical.
  4. Finally, the use of one-time symmetric keys strengthens what is already a strong symmetric encryption approach. Only a small amount of plaintext is encrypted with each key, and there is no relationship among the keys. Thus, to the extent that the public-key algorithm is secure, the entire scheme is secure.
  5. PGP provides the user with a range of key size options from 768 to 3072 bits .

Confidentiality and Authentication

- Both services may be used for the same message. First, a signature is generated for the plaintext Message and prepended to the message.
- Then the plaintext message plus signature is encrypted using CAST-128 or IDEA or 3DES, and the session key is encrypted using RSA or ElGamal.
- More convenient to store a signature with a plaintext version of a message.
- For purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.
- When both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and finally encrypts the session key with the recipient's public key.

Compression

- PGP compresses the message after applying the signature but before encryption, saving space both for e-mail transmission and for file storage.
- Indicated by Z for compression and Z-1 for decompression.
- The signature is generated before compression for two reasons:

  a. preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

  b. Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms.

  However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.
- Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.
- The compression algorithm used is ZIP.

E-mail Compatibility

1. Many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters.
2. The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors.
3. The use of radix 64 expands a message by 33%. The session key and signature portions of the message are relatively compact, and the plaintext message has been compressed. The compression should be more than enough to compensate for the radix-64 expansion.
4. It blindly converts the input stream to radix-64 format regardless of content, even if the input happens to be ASCII text.

5. PGP can be configured to convert to radix-64 format only the signature portion of signed plaintext messages. This enables the human recipient to read the message without using PGP.

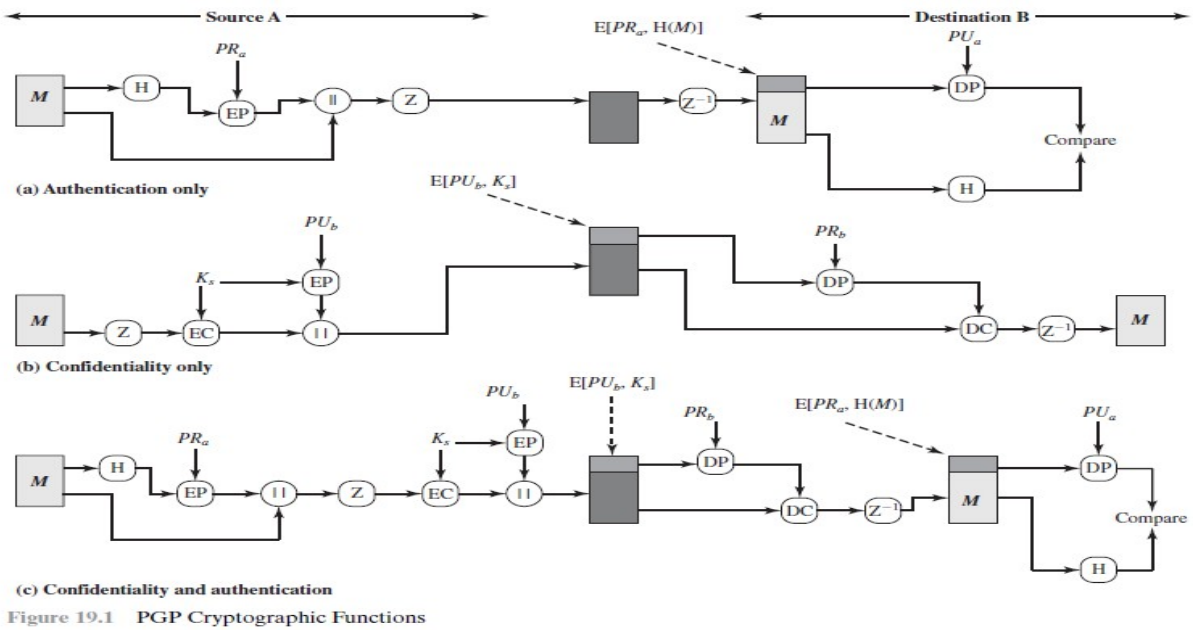
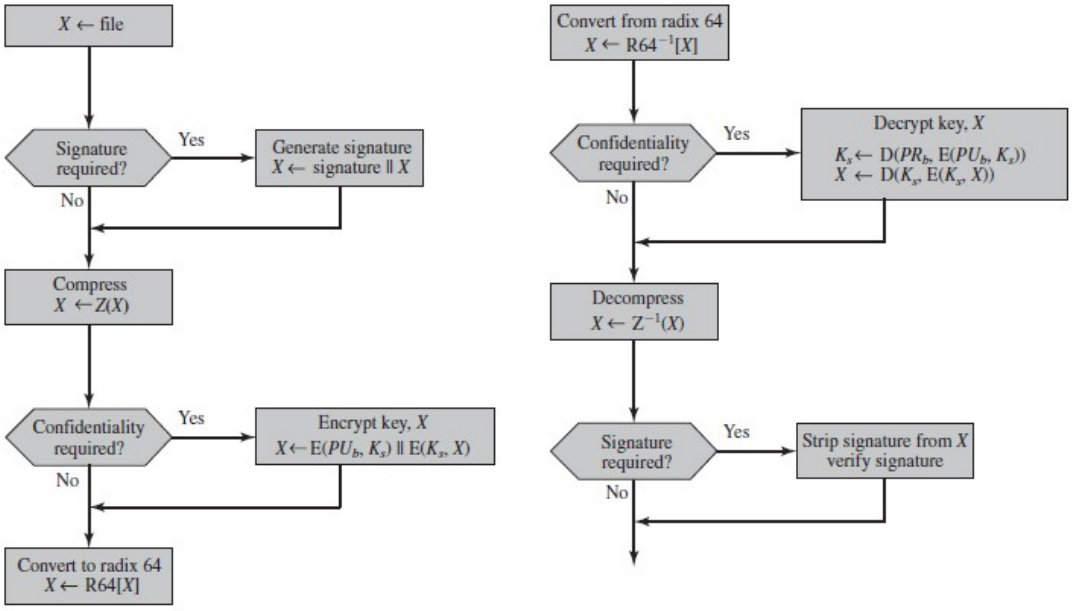


Figure 19.1 PGP Cryptographic Functions

PGP OPERATION

1. On transmission, a signature is generated using a hash code of the uncompressed plaintext.
2. Then the plaintext (plus signature if present) is compressed.
3. Next, if confidentiality is required, the block (compressed plaintext or compressed signature plus plaintext) is encrypted and prepended with the public key-encrypted symmetric encryption key.
4. Finally, the entire block is converted to radix-64 format.
5. On reception, the incoming block is first converted back from radix-64 format to binary.
6. Then, if the message is encrypted, the recipient recovers the session key and decrypts the message. The resulting block is then decompressed. If the message is signed, the recipient recovers the transmitted hash code and compares it to its own calculation of the hash code.



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Figure 19.2 Transmission and Reception of PGP Messages

MIME

RFC 822 defines a format for text messages that are sent using e-mail. The format adds to the text message an envelope of meta-data, namely the header. The header is separated from the body of the email by a blank line. Each line in the header consists of a keyword such as *From*, *To*, *Subject*, *Date*. Mime addresses shortcomings in the 822 scheme or the Simple Mail Transfer protocol. For instance

- SMTP cannot transmit executable or other binary data. (But UNIX users can use uuencode / uudecode.)
- SMTP text is confined to the 128 7-bit ASCII characters.
- SMTP servers can limit the size of email.
- Not all SMTP implementations adhere completely to the SMTP standard. Problems are with the treatment of carriage return and linefeeds, truncating or wrapping lines longer than 76 characters, padding lines, and conversion of tab characters.
- SMTP servers may reject mail message over a certain size.
- SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

- SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.

MIME introduces five new headers.

- MIME-Version: This field must have a parameters value of 1.0 to indicate that the message conforms to RFC 2045 and 2046.
- Content-Type: Describes the data contained in the body so that the receiver can pick the appropriate application to represent the data to the user.
- Content-Transfer-Encoding: Indicates the type of transformation that has been used to represent the body of the message in order to render it amenable to the mail tranport.
- Content-ID: Used to identify MIME entities.
- Content-Description: A text description of the object within the body, e.g. audio-data.

S/MIME

- After the development of PEM industry working group led by RSA Security, Inc. started to develop another specification for conveying digitally signed and/or encrypted and digitally enveloped data in accordance to the MIME message formats.

- S/MIME (**Secure/Multipurpose Internet Mail Extension**) is a security enhancement to the MIME Internet e-mail format standard.

- S/MIME is ***not restricted to mail***; it can be used with any transport mechanism that transports MIME data, such as HTTP.

- S/MIME is ***likely to emerge as the industry standard*** for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many.
- S/MIME provides the following cryptography security services:

1. Authentication.
2. Message Integrity. By using digital signing
3. Non-repudiation of origin.
4. Privacy and data security. By using encryption

There are three versions of S/MIME:

- S/MIME version **1 (1995)**- was specified and officially published in 1995 by RSA Security, Inc.
- S/MIME version **2 (1998)**- was specified in a pair of informational RFC documents - RFC 2311 and RFC 2312 - in March1998.

- The work was continued in the IETF S/MIME Mail Security (SMIME) WG and resulted in S/MIME version **3 (1999)** specified in RFCs 2630 to 2634 in June 1999.

Table 19.2 MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

Table 19.3 MIME Transfer Encodings

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present, but the lines are not neces-sarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

The other major component of MIME is a definition of transfer encodings for message contents:***Canonical Form*** **An important concept in MIME and S/MIME is that of canonical** form. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in contrast to native form, which is a format that may be peculiar to a particular system.

S/MIME has the same functionality as PGP, that is, it offers the ability to sign and to encrypt messages. In more detail, S/MIM provides:

- Enveloped Data, to apply privacy protection to a message. A sender needs to have access to a public key for each intended message recipient.
- Signed Data, to provide authentication. Only a S/MIME enabled mailer can view this message.
- Clear-signed Data, to provide authentication for users with S/MIME capabilities, but to retain readability other viewers.
- Nesting of signed and encrypted data.

Cryptographic Algorithms Table 19.5 summarizes the cryptographic algorithms used in S/MIME. S/MIME uses the following terminology taken from RFC 2119 (*Key Words for use in RFCs to Indicate Requirement Levels*) to specify the requirement level:

• **MUST:** The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.

• **SHOULD:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

S/MIME incorporates three public-key algorithms, DSS for digital sigantures, Diffie-Hellman for encyrpting session keys, or RSA. It uses SHA1 or MD5 for calculating digests, and three-key triple DES for message encryption. In an ideal situation, a S/MIME sender has a list of preferred decrypting capabilities from an intended recipient, in which case it chooses the best encryption. Otherwise, if the sender has received any previous mail from the intended recipient, it then chooses the same encryption mechanism.

To secure a MIME entity (e.g. the entire message with exception of the RFC 822 header), S/MIME produces a PKCS object. The PKCS object is then treated as the message object and encoded with MIME. Since the result of encryption is typically in binary, it needs to be transferred in a more secure way, such as in base64 mode.

To make an EnvelopedData MIME entity we:

- generate a pseudo-random session key for either tripleDES or RC2/40 (a weak, exportable encryption).
- for each recipient, encrypt the session key with the recipients public RSA key.
- for each recipient, prepare a block known as RecipientInfo that contains the sender's public-key certificate, an identifier for the algorithm used to encrypt the session key, and the encrypted session key.
- encrypt the message content with the session key.

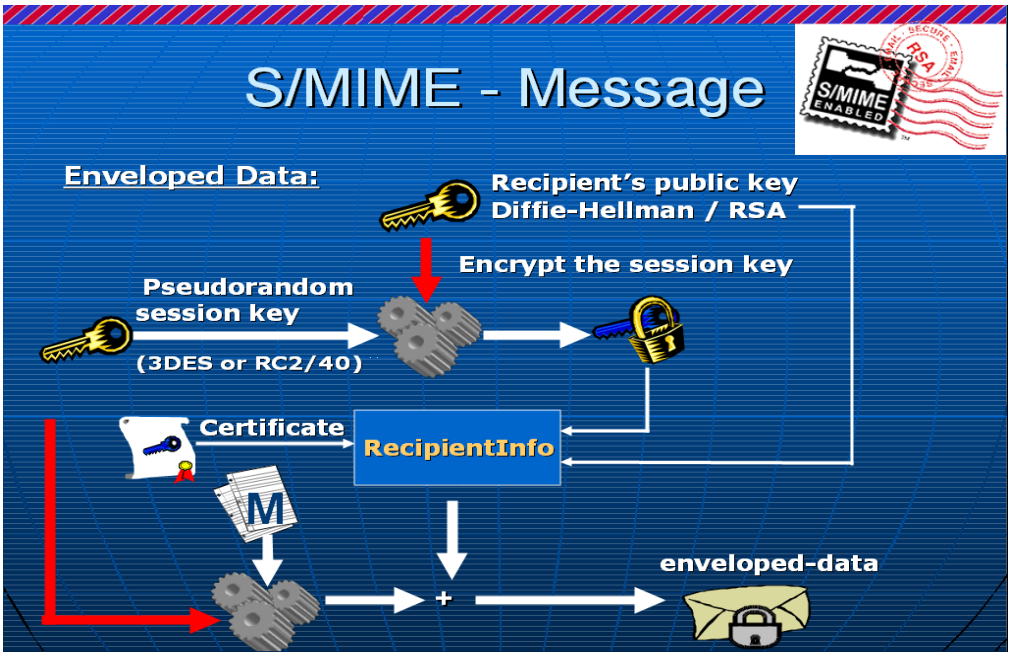
To recover the encrypted message, the recipient first reconverts the base64 encoding and uses his

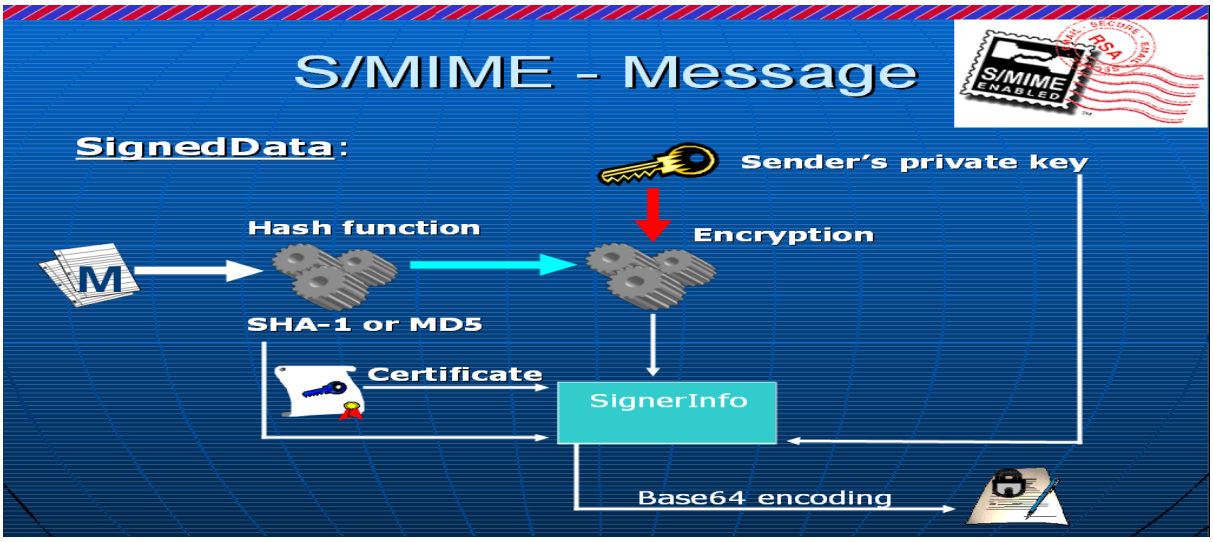
private key to recover the session key. He uses this key to decrypt the message.

To make an SignedData MIME entity we:

- select either SHA1 or MD5
- compute the message digest of the content to be signed
- encrypt the message digest with the signer's private key
- prepare the SignerIno block that contains the signer's public key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.
- the whole block is then encoded in to base64 (excluding the RFC 822 header).

Clear signing uses the multipart content type in MIME to transmit body and signature separately. The body needs to be encoded in some way so that it is not altered during transit. The second object, the signature, is sent in base64. Thus, only S/MIME enabled mail readers can use the signature.





Certificate-only message:

- Used to transport certificates.
- contains only certificates or a certificate revocation list (CRL).
- Sent in response to a registration request.
- The message is an application/pkcs7-mime type/subtype.

Creating a Certificates-only Message:

Step 1:

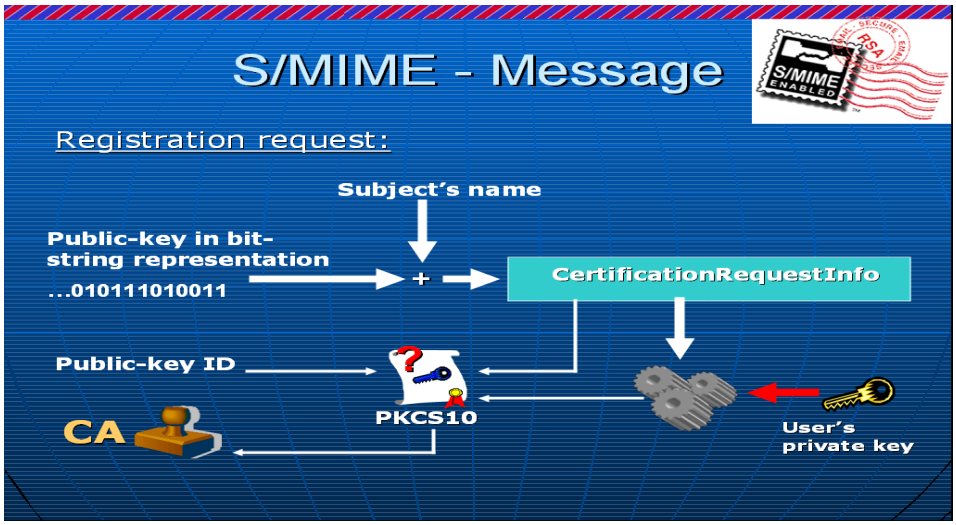
The certificates are made available to the CMS generating process which creates a CMS object of
type signedData.

Step 2:

The CMS signedData object is enclosed in an application/pkcs7-mime MIME entity.
The smime-type parameter for a certs-only message is "certs-only".
The file extension for this type of message is ".p7c".

Registration request:

- A message signer MUST have a certificate for the signature so that the receiving agent can verify the signature.
- Exchange with CA, hardware token, diskette etc.
- S/MIME v3- does not specify how to request a certification.
- S/MIME v2- request by applying to a CA using the application/pkcs10 S/MIME entity.
- A typical app. Only needs to send certification request.



S/MIME Messages

S/MIME makes use of a number of new MIME content types, which are shown in Table 19.6. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

Table 19.6 S/MIME Content Types

Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs7-mime	signedData	A signed S/MIME entity.
	pkcs7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs7-mime	degenerate signedData	An entity containing only public-key certificates.
	pkcs7-mime	CompressedData	A compressed S/MIME entity.
	pkcs7-signature	signedData	The content type of the signature subpart of a multipart/signed message.

S/MIME - Certificates

S/MIME uses public-key certificates that conform to version 3 of X.509.

A hybrid between a strict X.509 certification hierarchy and PGP's web of trust.

A receiving agent MUST provide some certificate retrieval mechanism.

Receiving and sending agents SHOULD also provide a mechanism to allow a user to "store and protect" certificates

Public key certificates are required to protect the authenticity and integrity of public keys, thus protecting against man-in-the-middle attack.

A certificate chain must be verified until a **root CA is reached**

a certificate can only be trusted if:
every certificate in the chain is successfully verified.
every CA in the certificate chain is trusted.
In practice, certificate chains are short and seldom verified for trustworthiness.
Also, the concept of cross-certification is of low practical value and seldom used between certification service providers.

Key generation:
MUST be capable of generating separate Diffie-Hellman and DSS key pairs.
SHOULD be capable of generating RSA key pairs.
good source of non-deterministic random.
protected in a secure fashion.

Registration:
A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.

Certificate storage and retrieval:
access to a local list of certificates in order to verify incoming signatures and encrypt outgoing messages.
maintained by the user local administrative entity on behalf of number of users.

Certificate Management in S/MIME:
CA-centered.
CA certificates come with the client software.
An ordinary user is not aware of the CAs that he/she trusts.
Certificates are sent along with the signed messages.
Certificates classes (common practice by most CAs)
– Class 1
– Class 2
– Class 3
CA certification policies
– ID-control practices
• Class 1: only email address
• Class 2: against third party database
• Class 3: apply in person and submit picture IDs and/or hard documentation

UNIT-V

Security at the Transport layer:

Network security entails securing data against attacks while it is in transit on a network. To achieve this goal, many real-time security protocols have been designed. There are popular standards for real-time network security protocols such as S/MIME, SSL/TLS, SSH, and IPsec. As mentioned earlier, these protocols work at different layers of networking model.

In the last chapter, we discussed some popular protocols that are designed to provide application layer security. In this chapter, we will discuss the process of achieving network security at Transport Layer and associated security protocols.

For TCP/IP protocol based network, physical and data link layers are typically implemented in the user terminal and network card hardware. TCP and IP layers are implemented in the operating system. Anything above TCP/IP is implemented as user process.

Need for Transport Layer Security

Let's discuss a typical Internet-based business transaction.

Bob visits Alice’s website for selling goods. In a form on the website, Bob enters the type of good and quantity desired, his address and payment card details. Bob clicks on Submit and waits for delivery of goods with debit of price amount from his account. All this sounds good, but in absence of network security, Bob could be in for a few surprises.

- If transactions did not use confidentiality (encryption), an attacker could obtain his payment card information. The attacker can then make purchases at Bob's expense.
- If no data integrity measure is used, an attacker could modify Bob's order in terms of type or quantity of goods.
- Lastly, if no server authentication is used, a server could display Alice's famous logo but the site could be a malicious site maintained by an attacker, who is masquerading as Alice. After receiving Bob's order, he could take Bob's money and flee. Or he could carry out an identity theft by collecting Bob's name and credit card details.

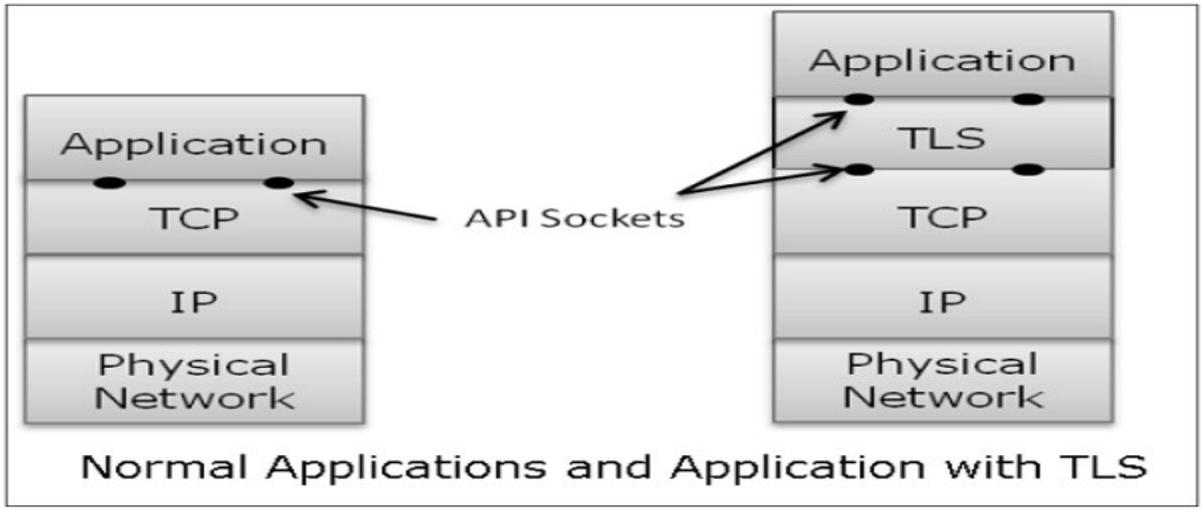
Transport layer security schemes can address these problems by enhancing TCP/IP based network communication with confidentiality, data integrity, server authentication, and client authentication.

The security at this layer is mostly used to secure HTTP based web transactions on a network. However, it can be employed by any application running over TCP.

Philosophy of TLS Design

Transport Layer Security (TLS) protocols operate above the TCP layer. Design of these protocols use popular Application Program Interfaces (API) to TCP, called “sockets" for interfacing with TCP layer.

Applications are now interfaced to Transport Security Layer instead of TCP directly. Transport Security Layer provides a simple API with sockets, which is similar and analogous to TCP's API.



In the above diagram, although TLS technically resides between application and transport layer, from the common perspective it is a transport protocol that acts as TCP layer enhanced with security services.

TLS is designed to operate over TCP, the reliable layer 4 protocol (not on UDP protocol), to make design of TLS much simpler, because it doesn't have to worry about ‘timing out’ and ‘retransmitting lost data’. The TCP layer continues doing that as usual which serves the need of TLS.

Why TLS is Popular?

The reason for popularity of using a security at Transport Layer is simplicity. Design and deployment of security at this layer does not require any change in TCP/IP protocols that are implemented in an operating system. Only user processes and applications needs to be designed/modified which is less complex.

Secure Socket Layer (SSL)

In this section, we discuss the family of protocols designed for TLS. The family includes SSL versions 2 and 3 and TLS protocol. SSLv2 has been now replaced by SSLv3, so we will focus on SSL v3 and TLS.

Brief History of SSL

In year 1995, Netscape developed SSLv2 and used in Netscape Navigator 1.1. The SSL version1 was never published and used. Later, Microsoft improved upon SSLv2 and introduced another similar protocol named Private Communications Technology (PCT).

Netscape substantially improved SSLv2 on various security issues and deployed SSLv3 in 1999. The Internet Engineering Task Force (IETF) subsequently, introduced a similar TLS (Transport Layer Security) protocol as an open standard. TLS protocol is non-interoperable with SSLv3.

TLS modified the cryptographic algorithms for key expansion and authentication. Also, TLS suggested use of open crypto Diffie-Hellman (DH) and Digital Signature Standard (DSS) in place of patented RSA crypto used in SSL. But due to expiry of RSA patent in 2000, there existed no strong reasons for users to shift away from the widely deployed SSLv3 to TLS.

Salient Features of SSL

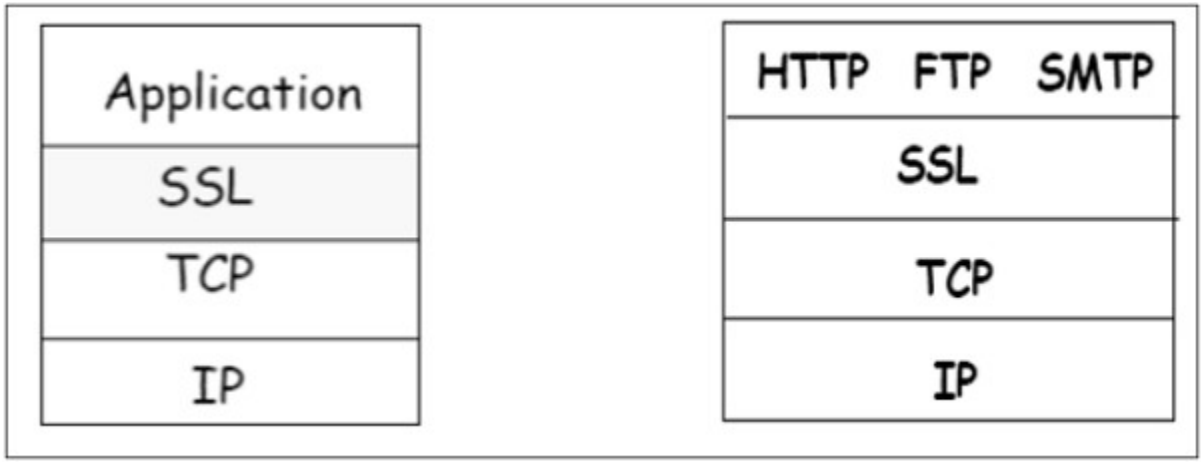
The salient features of SSL protocol are as follows −

- SSL provides network connection security through −
    - **Confidentiality** − Information is exchanged in an encrypted form.
    - **Authentication** − Communication entities identify each other through the use of digital certificates. Web-server authentication is mandatory whereas client authentication is kept optional.
    - **Reliability** − Maintains message integrity checks.
- SSL is available for all TCP applications.
- Supported by almost all web browsers.
- Provides ease in doing business with new online entities.
- Developed primarily for Web e-commerce.

Architecture of SSL

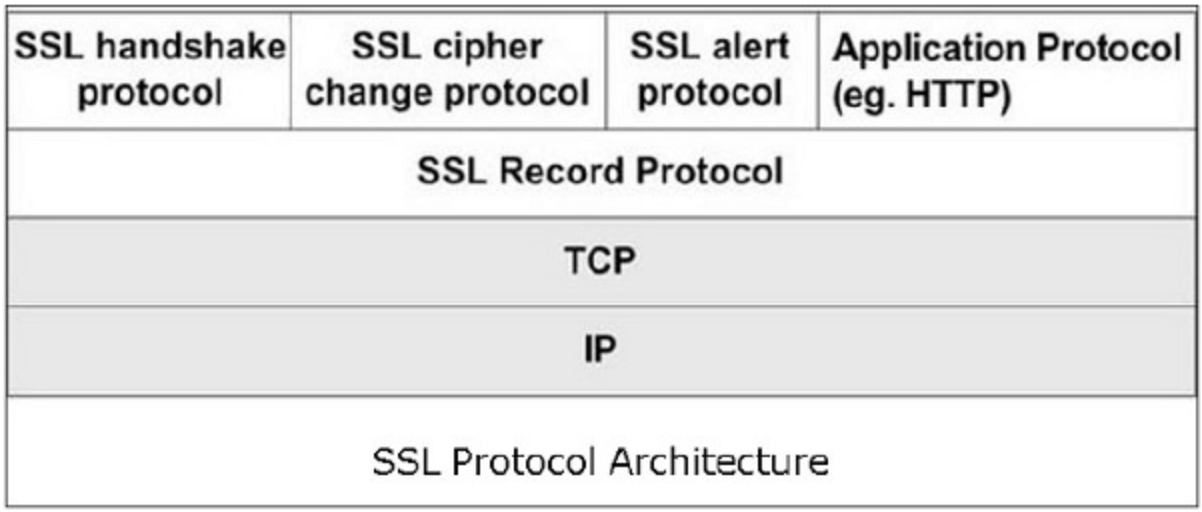
SSL is specific to TCP and it does not work with UDP. SSL provides Application Programming Interface (API) to applications. C and Java SSL libraries/classes are readily available.

SSL protocol is designed to interwork between application and transport layer as shown in the following image −



SSL itself is not a single layer protocol as depicted in the image; in fact it is composed of two sub-layers.

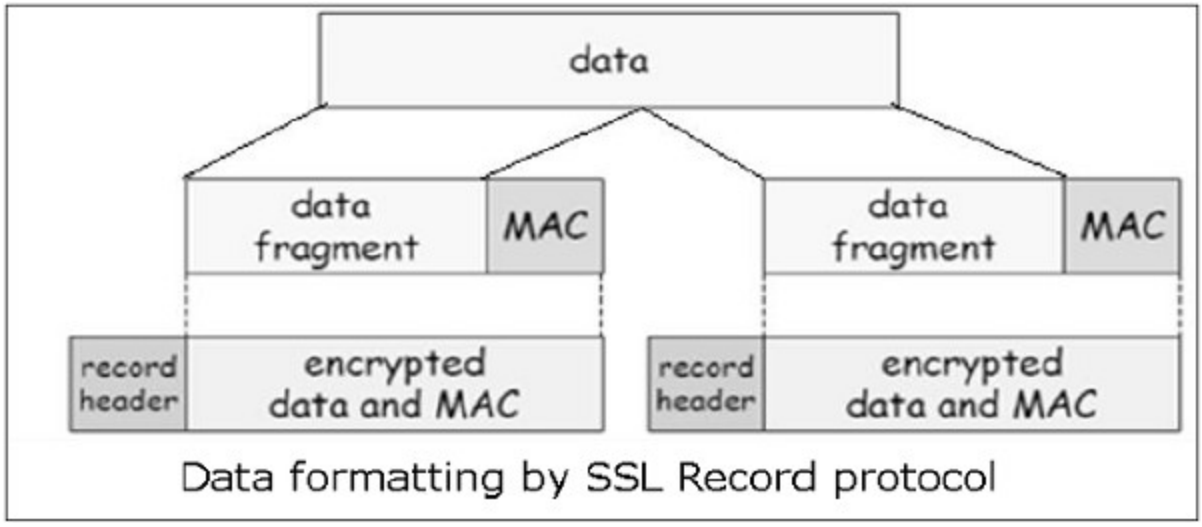
- Lower sub-layer comprises of the one component of SSL protocol called as SSL Record Protocol. This component provides integrity and confidentiality services.
- Upper sub-layer comprises of three SSL-related protocol components and an application protocol. Application component provides the information transfer service between client/server interactions. Technically, it can operate on top of SSL layer as well. Three SSL related protocol components are −
    - SSL Handshake Protocol
    - Change Cipher Spec Protocol
    - Alert Protocol.
- These three protocols manage all of SSL message exchanges and are discussed later in this section.



Functions of SSL Protocol Components

The four sub-components of the SSL protocol handle various tasks for secure communication between the client machine and the server.

- Record Protocol
    - The record layer formats the upper layer protocol messages.
    - It fragments the data into manageable blocks (max length 16 KB). It optionally compresses the data.
    - Encrypts the data.
    - Provides a header for each message and a hash (Message Authentication Code (MAC)) at the end.
    - Hands over the formatted blocks to TCP layer for transmission.



Data formatting by SSL Record protocol

- SSL Handshake Protocol
    - It is the most complex part of SSL. It is invoked before any application data is transmitted. It creates SSL sessions between the client and the server.
    - Establishment of session involves Server authentication, Key and algorithm negotiation, Establishing keys and Client authentication (optional).
    - A session is identified by unique set of cryptographic security parameters.
    - Multiple secure TCP connections between a client and a server can share the same session.
    - Handshake protocol actions through four phases. These are discussed in the next section.
- ChangeCipherSpec Protocol

- Simplest part of SSL protocol. It comprises of a single message exchanged between two communicating entities, the client and the server.
- As each entity sends the ChangeCipherSpec message, it changes its side of the connection into the secure state as agreed upon.
- The cipher parameters pending state is copied into the current state.
- Exchange of this Message indicates all future data exchanges are encrypted and integrity is protected.

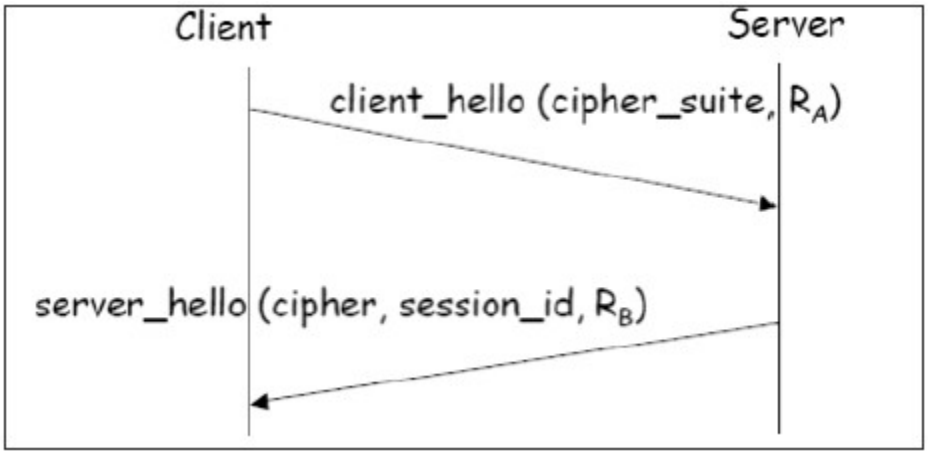
- SSL Alert Protocol
  - This protocol is used to report errors – such as unexpected message, bad record MAC, security parameters negotiation failed, etc.
  - It is also used for other purposes – such as notify closure of the TCP connection, notify receipt of bad or unknown certificate, etc.

Establishment of SSL Session

As discussed above, there are four phases of SSL session establishment. These are mainly handled by SSL Handshake protocol.

Phase 1 − Establishing security capabilities.

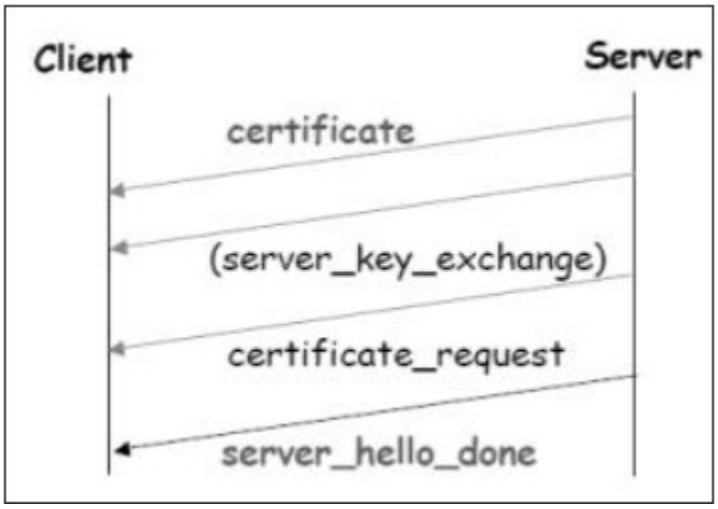
- This phase comprises of exchange of two messages – *Client_hello* and *Server_hello*.



- *Client_hello* contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.
- *Server_hello* contains the selected Cipher Specification (CipherSpec) and a new *session_id*.
- The CipherSpec contains fields like −
  - Cipher Algorithm (DES, 3DES, RC2, and RC4)

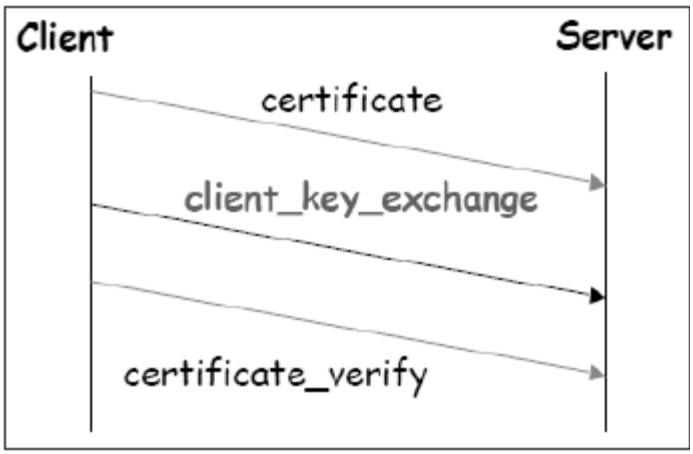
- MAC Algorithm (based on MD5, SHA-1)
- Public-key algorithm (RSA)
- Both messages have “nonce” to prevent replay attack.

Phase 2 − Server authentication and key exchange.



- Server sends certificate. Client software comes configured with public keys of various “trusted” organizations (CAs) to check certificate.
- Server sends chosen cipher suite.
- Server may request client certificate. Usually it is not done.
- Server indicates end of *Server_hello*.

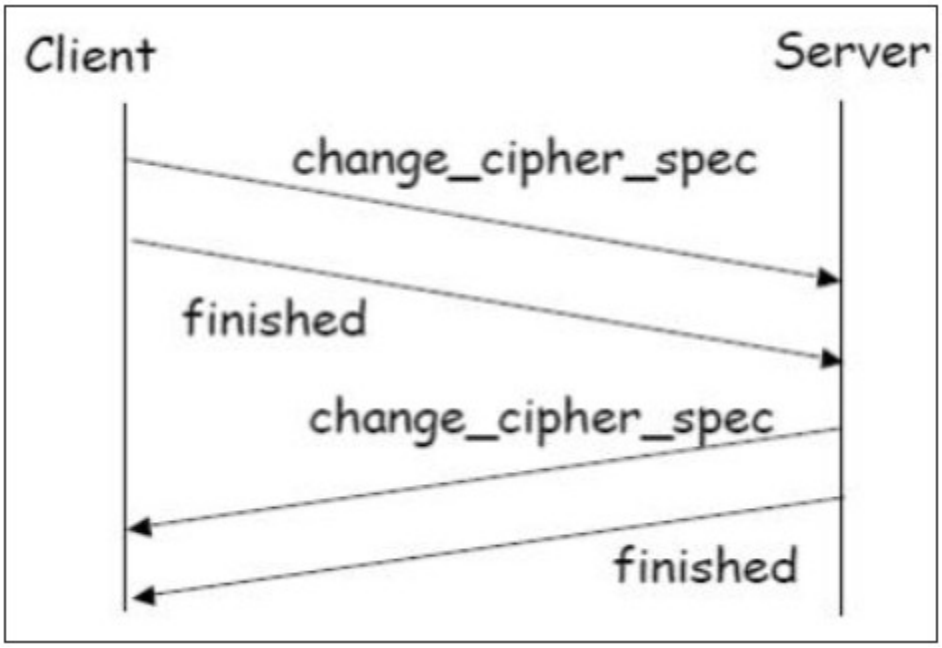
Phase 3 − Client authentication and key exchange.



- Client sends certificate, only if requested by the server.
- It also sends the Pre-master Secret (PMS) encrypted with the server’s public key.

- Client also sends *Certificate_verify* message if certificate is sent by him to prove he has the private key associated with this certificate. Basically, the client signs a hash of the previous messages.

Phase 4 − Finish.



- Client and server send *Change_cipher_spec* messages to each other to cause the pending cipher state to be copied into the current state.
- From now on, all data is encrypted and integrity protected.
- Message “Finished” from each end verifies that the key exchange and authentication processes were successful.

All four phases, discussed above, happen within the establishment of TCP session. SSL session establishment starts after TCP SYN/ SYNACK and finishes before TCP Fin.

Resuming a Disconnected Session

- It is possible to resume a disconnected session (through *Alert* message), if the client sends a *hello_request* to the server with the encrypted *session_id* information.
- The server then determines if the *session_id* is valid. If validated, it exchanges ChangeCipherSpec and *finished* messages with the client and secure communications resume.
- This avoids recalculating of session cipher parameters and saves computing at the server and the client end.

SSL Session Keys

We have seen that during Phase 3 of SSL session establishment, a pre-master secret is sent by the client to the server encrypted using server's public key. The master secret and various session keys are generated as follows −

- The master secret is generated (via pseudo random number generator) using −
  - The pre-master secret.
  - Two nonces (RA and RB) exchanged in the client_hello and server_hello messages.
- Six secret values are then derived from this master secret as −
  - Secret key used with MAC (for data sent by server)
  - Secret key used with MAC (for data sent by client)
  - Secret key and IV used for encryption (by server)
  - Secret key and IV used for encryption (by client)

TLS Protocol

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

Salient Features

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

Comparison of TLS and SSL Protocols

There are main eight differences between TLS and SSLv3 protocols. These are as follows −

- **Protocol Version** − The header of TLS protocol segment carries the version number 3.1 to differentiate between number 3 carried by SSL protocol segment header.

- **Message Authentication** − TLS employs a keyed-hash message authentication code (H-MAC). Benefit is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.
- **Session Key Generation** − There are two differences between TLS and SSL protocol for generation of key material.
    - Method of computing pre-master and master secrets is similar. But in TLS protocol, computation of master secret uses the HMAC standard and pseudorandom function (PRF) output instead of ad-hoc MAC.
    - The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.
- Alert Protocol Message −
    - TLS protocol supports all the messages used by the Alert protocol of SSL, except *No certificate* alert message being made redundant. The client sends empty certificate in case client authentication is not required.
    - Many additional Alert messages are included in TLS protocol for other error conditions such as *record_overflow*, *decode_error* etc.
- **Supported Cipher Suites** − SSL supports RSA, Diffie-Hellman and Fortezza cipher suites. TLS protocol supports all suits except Fortezza.
- **Client Certificate Types** − TLS defines certificate types to be requested in a *certificate_request* message. SSLv3 support all of these. Additionally, SSL support certain other types of certificate such as Fortezza.
- CertificateVerify and Finished Messages −
    - In SSL, complex message procedure is used for the *certificate_verify* message. With TLS, the verified information is contained in the handshake messages itself thus avoiding this complex procedure.
    - Finished message is computed in different manners in TLS and SSLv3.
- **Padding of Data** − In SSL protocol, the padding added to user data before encryption is the minimum amount required to make the total data-size equal to a multiple of the cipher's block length. In TLS, the padding can be any amount that results in data-size that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

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

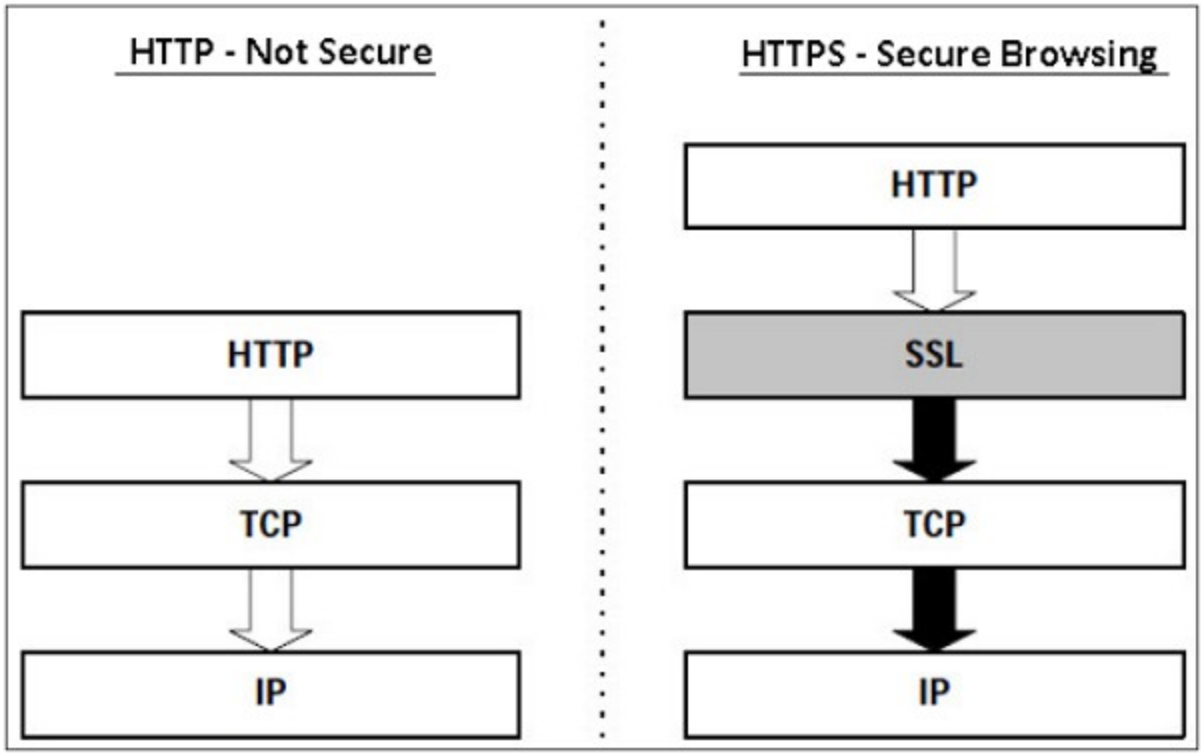
	SSL v3.0	TLS v1.0
Protocol version in messages	3.0	3.1
Alert protocol message types	12	23
Message authentication	ad hoc	standard
Key material generation	ad hoc	PRF
CertificateVerify	complex	simple
Finished	ad hoc	PRF
Baseline cipher suites	includes Fortezza	no Fortezza

Secure Browsing - HTTPS

In this section, we will discuss the use of SSL/TLS protocol for performing secure web browsing.

HTTPS Defined

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



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

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

Working of HTTPS

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

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

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

Use of HTTPS

- Use of HTTPS provides confidentiality, server authentication and message integrity to the user. It enables safe conduct of e-commerce on the Internet.
- Prevents data from eavesdropping and denies identity theft which are common attacks on HTTP.

Present day web browsers and web servers are equipped with HTTPS support. The use of HTTPS over HTTP, however, requires more computing power at the client and the server end to carry out encryption and SSL handshake.

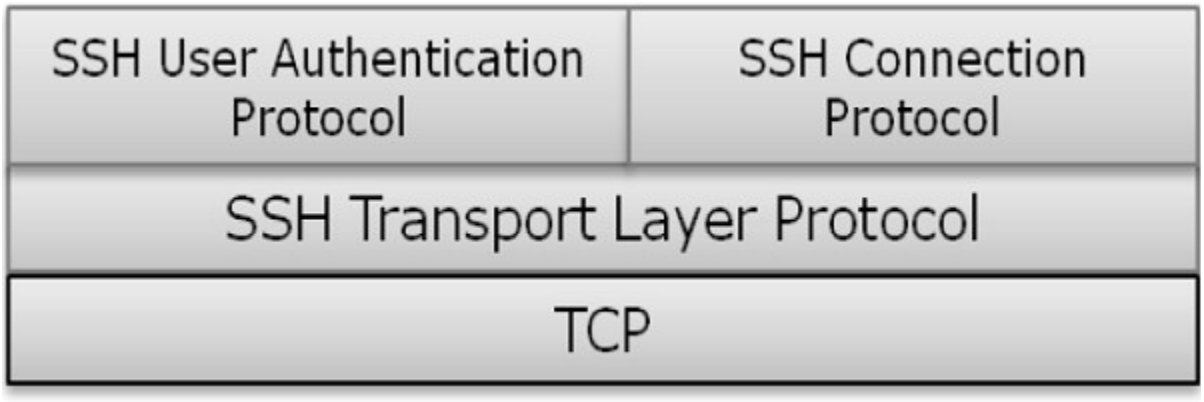
Secure Shell Protocol (SSH)

The salient features of SSH are as follows −

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

SSH Defined

SSH is organized as three sub-protocols.



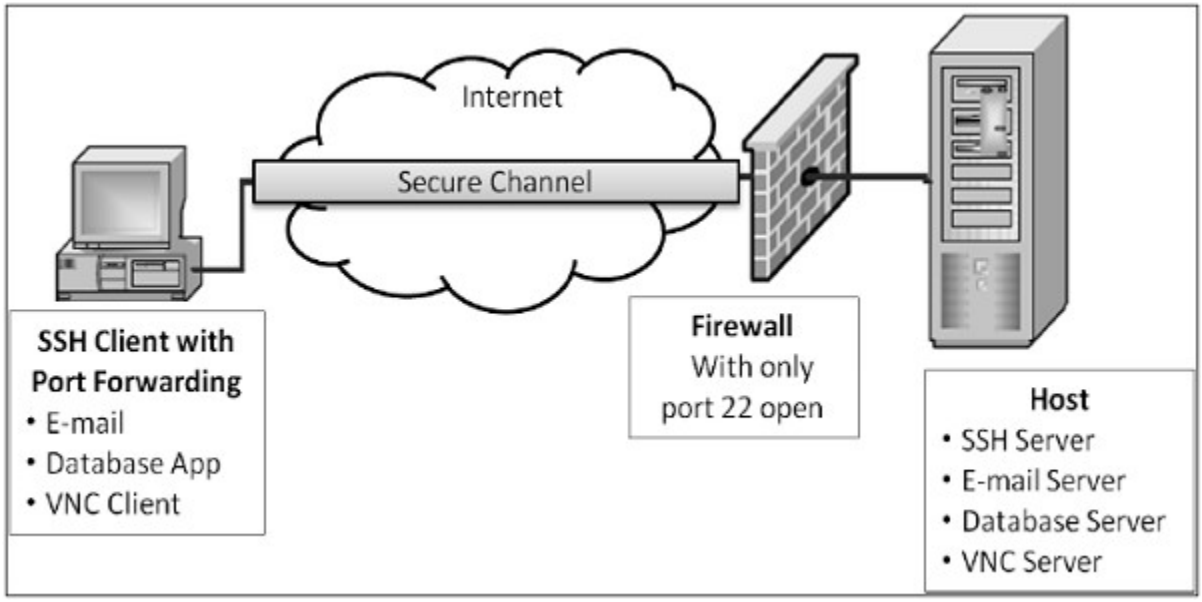
- **Transport Layer Protocol** − This part of SSH protocol provides data confidentiality, server (host) authentication, and data integrity. It may optionally provide data compression as well.
    - **Server Authentication** − Host keys are asymmetric like public/private keys. A server uses a public key to prove its identity to a client. The client verifies that contacted server is a “known” host from the database it maintains. Once the server is authenticated, session keys are generated.
    - **Session Key Establishment** − After authentication, the server and the client agree upon cipher to be used. Session keys are generated by both the client and the server. Session keys are generated before user authentication so that usernames and passwords can be sent encrypted. These keys are generally replaced at regular intervals (say, every hour) during the session and are destroyed immediately after use.

    - **Data Integrity** − SSH uses Message Authentication Code (MAC) algorithms to for data integrity check. It is an improvement over 32 bit CRC used by SSH1.
- **User Authentication Protocol** − This part of SSH authenticates the user to the server. The server verifies that access is given to intended users only. Many authentication methods are currently used such as, typed passwords, Kerberos, public-key authentication, etc.
- **Connection Protocol** − This provides multiple logical channels over a single underlying SSH connection.

SSH Services

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

- **Secure Command-Shell (Remote Logon)** − It allows the user to edit files, view the contents of directories, and access applications on connected device. Systems administrators can remotely start/view/stop services and processes, create user accounts, and change file/directories permissions and so on. All tasks that are feasible at a machine's command prompt can now be performed securely from the remote machine using secure remote logon.
- **Secure File Transfer** − SSH File Transfer Protocol (SFTP) is designed as an extension for SSH-2 for secure file transfer. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP encrypts both the username/password and the file data being transferred. It uses the same port as the Secure Shell server, i.e. system port no 22.
- **Port Forwarding (Tunneling)** − It allows data from unsecured TCP/IP based applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel to the program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed secure channel, eliminating the need to open many ports on a firewall or router.



Benefits & Limitations

The benefits and limitations of employing communication security at transport layer are as follows −

- Benefits
    - Transport Layer Security is transparent to applications.
    - Server is authenticated.
    - Application layer headers are hidden.
    - It is more fine-grained than security mechanisms at layer 3 (IPsec) as it works at the transport connection level.
- Limitations
    - Applicable to TCP-based applications only (not UDP).
    - TCP/IP headers are in clear.
    - Suitable for direct communication between the client and the server. Does not cater for secure applications using chain of servers (e.g. email)
    - SSL does not provide non-repudiation as client authentication is optional.
    - If needed, client authentication needs to be implemented above SSL.

IPSec

- Users have security concerns that cut across protocol layers.

- By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but also for the many security-ignorant applications.
- IP-level security encompasses three functional areas: authentication, confidentiality, and key management.
- The authentication mechanism assures that a received packet was, in fact, transmitted by the party identified as the source in the packet header. In addition, this mechanism assures that the packet has not been altered in transit.
- The confidentiality facility enables communicating nodes to encrypt messages to prevent eavesdropping by third parties. The key management facility is concerned with the secure exchange of keys.
- **Applications of IPsec**

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

    1. Secure branch office connectivity over the Internet
    2. Secure remote access over the Internet
    3. Establishing extranet and intranet connectivity with partners
    4. Enhancing electronic commerce security

6. The principal feature of IPsec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed applications **can be secured.**
7. **Benefits of IPsec**

    1. • When IPsec is implemented in a firewall or router, Traffic within a company or workgroup does not incur the overhead of security-related processing.

    2. • IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.

3. • IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router.

4. • IPsec can be transparent to end users. There is no need to train users on security mechanisms.

5. IPsec can provide security for individual users if needed.

Routing Applications

- IPsec can assure that

1. A router advertisement comes from an authorized router.
2. A neighbor advertisement comes from an authorized router.
3. A redirect message comes from the router to which the initial IP packet was sent.
4. A routing update is not forged.

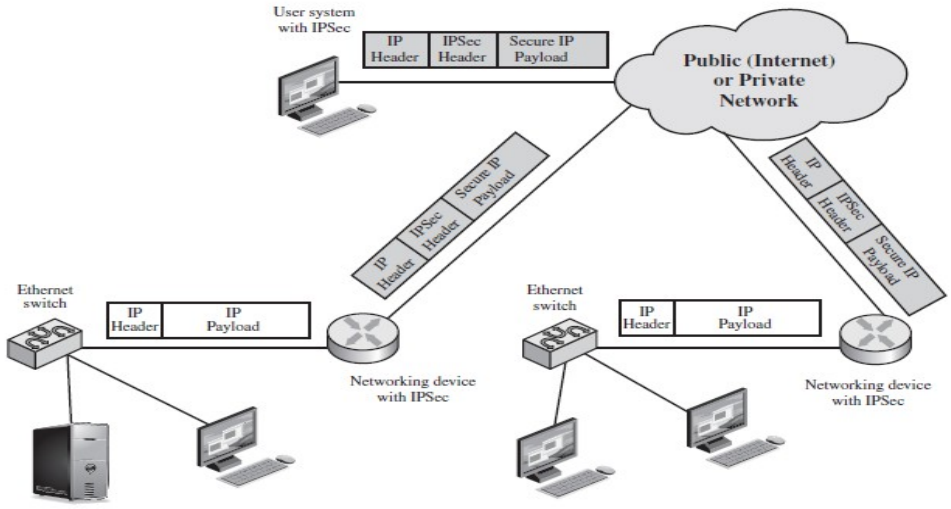


Figure 20.1 An IP Security Scenario

IPsec Services

5. IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services.

6. Two protocols are used to provide security: an authentication protocol designated by the header of the protocol, Authentication Header (AH); and a combined

encryption/ authentication protocol designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP).

7. RFC 4301 lists the following services:

    1. Access control
    2. Connectionless integrity
    3. Data origin authentication
    4. Rejection of replayed packets (a form of partial sequence integrity)
    5. Confidentiality (encryption)
    6. Limited traffic flow confidentiality

Modes of operation

IPsec can be implemented in a host-to-host transport mode, as well as in a network tunneling mode.

Transport Mode Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet.transport mode is used for endto- end communication between two hosts .

ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

Table 20.1 Tunnel Mode and Transport Mode Functionality

	Transport Mode SA	Tunnel Mode SA
AH	Authenticates IP payload and selected portions of IP header and IPv6 extension headers.	Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions of outer IP header and outer IPv6 extension headers.
ESP	Encrypts IP payload and any IPv6 extension headers following the ESP header.	Encrypts entire inner IP packet.
ESP with Authentication	Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IP payload but not IP header.	Encrypts entire inner IP packet. Authenticates inner IP packet.

Tunnel mode In tunnel mode, the entire IP packet is encrypted and authenticated. It is then encapsulated into a new IP packet with a new IP header. Tunnel mode is used to create virtual private networks for network-to-network communications , host-to-network communications and host-to-host communications (e.g. private chat).Tunnel mode supports NAT traversal.

- Tunnel mode is used when one or both ends of a security association (SA) are a security gateway, such as a firewall or router that implements IPsec. With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPsec.
- The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the Ipsec software in the firewall or secure router at the boundary of the local network.
- ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet, including the inner IP header. AH in tunnel mode authenticates the entire inner IP packet and selected portions of the outer IP header.
- A security association (SA) is a logical connection involving two devices that transfer data. With the help of the defined IPsec protocols, SAs offer data protection for unidirectional traffic. Generally, an IPsec tunnel features two unidirectional SAs, which offer a secure, full-duplex channel for data.
- A security association consists of features like traffic encryption key, cryptographic algorithm and mode, and also parameters required for the network data.
- In order to decide what protection is to be provided for an outgoing packet, IPsec uses the Security Parameter Index (SPI), an index to the security association database (SADB), along with the destination address in a packet header, which together uniquely identify a security association for that packet. A similar procedure is performed for an incoming packet, where IPsec gathers decryption and verification keys from the security association database.

For multicast, a security association is provided for the group, and is duplicated across all authorized receivers of the group. There may be more than one security association for a group, using different SPIs, thereby allowing multiple levels and sets of security within a group. Indeed, each sender can have multiple security associations, allowing authentication, since a receiver can only know that someone knowing the keys sent the data. Note that the relevant standard does not describe how the association is chosen and duplicated across the group; it is assumed that a responsible party will have made the choice.

Can be up to 32 bits large

The SPI allows the destination to select the correct SA under which the received packet

will be processed

According to the agreement with the sender

The SPI is sent with the packet by the sender

SPI + Dest IP address + IPSec Protocol (AH or ESP) uniquely identifies a SA

SA Database – SAD

- Holds parameters for each SA
- Lifetime of this SA
- AH and ESP information
- Tunnel or transport mode
- Every host or gateway participating in IPSec has their own SA database
- **Security Policies:** A *security policy* is a rule that is programmed into the IPSec implementation that tells it how to process different datagrams received by the device. For example, security policies are used to decide if a particular packet needs to be processed by IPSec or not; those that do not bypass AH and ESP entirely. If security is required, the security policy provides general guidelines for how it should be provided, and if necessary, links to more specific detail.

  Security policies for a device are stored in the device's *Security Policy Database (SPD)*.
- **Security Associations:** A *Security Association (SA)* is a set of security information that describes a particular kind of secure connection between one device and another. You can consider it a "contract", if you will, that specifies the particular security mechanisms that are used for secure communications between the two.

  A device's security associations are contained in its *Security Association Database (SAD)*.

It's often hard to distinguish the SPD and the SAD, since they are similar in concept. The main difference between them is that security policies are general while security associations are more specific. To determine what to do with a particular datagram, a device first checks the SPD. The security policies in the SPD may reference a particular security association in the SAD. If so, the

device will look up that security association and use it for processing the datagram.

The IPsec suite is an open standard. IPsec uses the following protocols to perform various functions:

- Authentication Headers (AH) provide connectionless data integrity and data origin authentication for IP datagrams and provides protection against replay attacks.
- Encapsulating Security Payloads (ESP) provide confidentiality, data-origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic-flow confidentiality.
- Security Associations (SA) provide the bundle of algorithms and data that provide the parameters necessary for AH and/or ESP operations. The Internet Security Association and Key Management Protocol (ISAKMP) provides a framework for authentication and key exchange,with actual authenticated keying material provided either by manual configuration with pre-shared keys, Internet Key Exchange (IKE and IKEv2), Kerberized Internet Negotiation of Keys (KINK), or IPSECKEY DNS records.

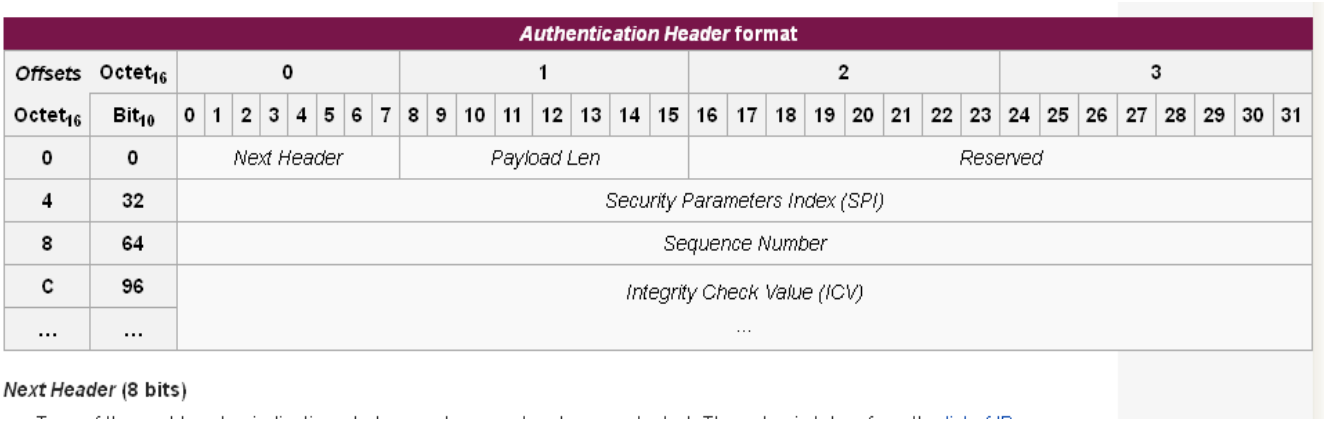
Authentication Header

Authentication Header (AH) is a member of the IPsec protocol suite. AH guarantees connectionless integrity and data origin authentication of IP packets. Further, it can optionally protect against replay attacks by using the sliding window technique and discarding old packets

- In IPv4, the AH protects the IP payload and all header fields of an IP datagram except for mutable fields (i.e. those that might be altered in transit), and also IP options such as the IP Security Option (RFC 1108). Mutable (and therefore unauthenticated) IPv4 header fields are DSCP/ToS, ECN, Flags, Fragment Offset, TTL and Header Checksum.
- In IPv6, the AH protects most of the IPv6 base header, AH itself, non-mutable extension headers after the AH, and the IP payload. Protection for the IPv6 header excludes the mutable fields: DSCP, ECN, Flow Label, and Hop Limit.

AH operates directly on top of IP, using IP protocol number 51.

ext Header (8 bits)



Authentication Header format

Offsets	Octet$_{16}$	0								1								2								3							
Octet$_{16}$	Bit$_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Next Header								Payload Len								Reserved															
4	32	Security Parameters Index (SPI)																															
8	64	Sequence Number																															
C	96	Integrity Check Value (ICV)																															
...	...	...																															

Next Header (8 bits)

Type of the next header, indicating what upper-layer protocol was protected. The value is taken from the list of IP protocol numbers.

Payload Len (8 bits)

The length of this *Authentication Header* in 4-octet units, minus 2. For example, an AH value of 4 equals 3×(32-bit fixed-length AH fields) + 3×(32-bit ICV fields) − 2 and thus an AH value of 4 means 24 octets. Although the size is measured in 4-octet units, the length of this header needs to be a multiple of 8 octets if carried in an IPv6 packet. This restriction does not apply to an *Authentication Header* carried in an IPv4 packet.

Reserved (16 bits)

Reserved for future use (all zeroes until then).

Security Parameters Index (32 bits)

Arbitrary value which is used (together with the destination IP address) to identify the security association of the receiving party.

Sequence Number (32 bits)

A monotonic strictly increasing sequence number (incremented by 1 for every packet sent) to prevent replay attacks. When replay detection is enabled, sequence numbers are never reused, because a new security association must be renegotiated before an attempt to increment the sequence number beyond its maximum value.

Integrity Check Value (multiple of 32 bits)

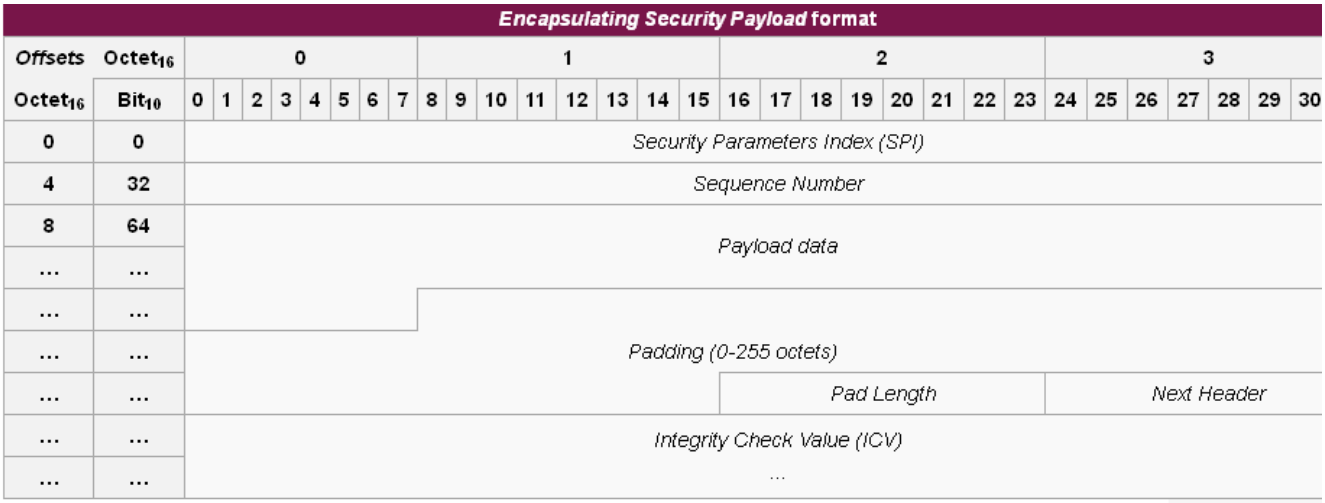
Variable length check value. It may contain padding to align the field to an 8-octet boundary for IPv6, or a 4-octet boundary for IPv4.

Encapsulating Security Payload

Encapsulating Security Payload (ESP) is a member of the IPsec protocol suite. In IPsec it provides origin authenticity, integrity and confidentiality protection of packets. ESP also supports encryption-only and authentication-only configurations, but using encryption without authentication is strongly discouraged because it is insecure. Unlike Authentication Header (AH), ESP in transport mode does not provide integrity and authentication for the entire IP packet. However, in Tunnel Mode, where the entire original IP packet is encapsulated with a new packet header added, ESP protection is afforded to the whole inner IP packet (including the inner

header) while the outer header (including any outer IPv4 options or IPv6 extension headers) remains unprotected. ESP operates directly on top of IP, using IP protocol number 50.

The following ESP packet diagram shows how an ESP packet is constructed and interpreted:



Security Parameters Index (32 bits)
: Arbitrary value used (together with the destination IP address) to identify the security association of the receiving party.

Sequence Number (32 bits)
: A monotonically increasing sequence number (incremented by 1 for every packet sent) to protect against replay attacks. There is a separate counter kept for every security association.

Payload data (variable)
: The protected contents of the original IP packet, including any data used to protect the contents (e.g. an Initialisation Vector for the cryptographic algorithm). The type of content that was protected is indicated by the *Next Header* field.

Padding (0-255 octets)
: Padding for encryption, to extend the payload data to a size that fits the encryption's cipher block size, and to align the next field.

Pad Length (8 bits)
: Size of the padding (in octets).

Next Header (8 bits)
: Type of the next header. The value is taken from the list of IP protocol numbers.

Integrity Check Value (multiple of 32 bits)
: Variable length check value. It may contain padding to align the field to an 8-octet boundary for IPv6, or a 4-octet boundary for IPv4.

- Hides the identity of your network
- Provides secure channel: confidentiality, authenticity, and integrity
- Connects sites (e.g., branch offices) with a cost- effective secure network compared with leased lines
- Allows user to work from home and mobile hosts

IPSec Cons

- A single failure in the path disconnect the entire network. Also cause performance bottlenecks.
- Incompatible with NAT/PAT depending on the architecture
- Tunneled traffic is undetected by IDS
- VPN gateways might be compromised which leads to uncovering protected data

---

Firewall:

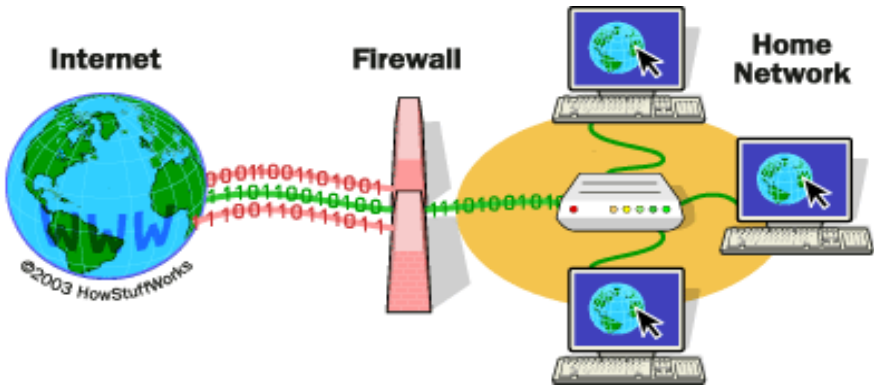
What is a Firewall?

-A firewall is hardware or software (or a combination of hardware and software) that **monitors the transmission of packets** of digital information that attempt to pass through the perimeter of a network.

-A firewall is simply a program or hardware device that filters the information coming through the Internet connection into your private network or computer system. If an incoming packet of

information is flagged by the filters, it is not allowed through.

Perimeter Defense



-A firewall is said to provide “perimeter security” because it sits on the outer boundary, or perimeter, of a network. The network boundary is the point at which one network connects to another.

Role of a Firewall?

i)a **choke point** that keeps unauthorized users out of the protected network.

-Firewalls create checkpoints (or choke points) between an internal private network and an untrusted Internet. Once the choke points have been clearly established, the device can monitor, filter and verify all inbound and outbound traffic.

ii)interconnects networks with differing trust

iii)imposes restrictions on network services

iv)A firewall can limit network exposure by hiding the internal network systems and information from the public Internet.

iv)only authorized traffic is allowed

v)auditing and controlling access

-The firewall also enforces logging, and provides alarm capacities as well. By placing logging services at firewalls, security administrators can monitor all access to and from the Internet. Good logging strategies are one of the most effective tools for proper network security.

vi)can implement alarms for abnormal behavior

vii) The firewall provides protection from various kinds of IP spoofing and routing attacks. It can also serve as the platform for IPsec. Using the tunnel mode capability, the firewall can be used to implement Virtual Private Networks (VPNs). A VPN encapsulates all the encrypted data within an IP packet.

viii)provides **perimeter defence**

Firewall Limitations:

-cannot protect from attacks bypassing it

-cannot protect against internal threats

e.g. dissatisfied employee who cooperates with an external attacker.

-cannot protect against transfer of all virus infected programs or files,because of huge range of OS & file types

Firewall-Related Terminology

Bastion host:

A bastion host is a publicly accessible device for the network's security,which has a direct connection to a public network such as the Internet. The bastion host serves as a platform for any one of the three types of firewalls: packet filter, circuit-level gateway or application-level gateway. The bastion host's role falls into the following three common types:

• Single-homed bastion host: This is a device with only one network interface, normally used for an application-level gateway. The external router is configured to send all incoming data to the bastion host, and all internal clients are configured to send all outgoing data to the host.Accordingly, the host will test the data according to security guidelines.

* Dual-homed bastion host: This is a firewall device with at least two network interfaces. Dual-homed bastion hosts serve as application-level gateways and as packet filters and circuit-level

gateways as well. The advantage of using such hosts is that they create a complete break between the external network and the internal network. This break forces all incoming and outgoing traffic to pass through the host. The dual homed bastion host will prevent a security break-in when a hacker tries to access internal devices.

• Multihomed bastion host: Single-purpose or internal bastion hosts can be classified as either single-homed or multihomed bastion hosts. The latter are used to allow the user to enforce strict security mechanisms. When the security policy requires all inbound and outbound traffic to be sent through a proxy server, a new proxy server should be created for the new streaming application. On the new proxy server, it is necessary to implement strict security mechanisms such as authentication. When multihomed bastion hosts are used as internal bastion hosts, they must reside inside the organisation’s internal network, normally as application gateways that receive all incoming traffic from external bastion hosts.

They provide an additional level of security in case the external firewall devices are compromised. All the internal network devices are configured to communicate only with the internal bastion host.

Proxy Servers :(sometimes called firewalls) - that make network connections for you.:

Proxies are mostly used to control, or monitor, outbound traffic. Some application proxies cache the requested data. This lowers bandwidth requirements and decreases the access the same data for the next user. It also gives unquestionable evidence of what was transferred.There are two types of proxy servers.

1. Application Proxies - that do the work for you.

2. SOCKS Proxies - that cross wire ports.

Application Proxy

The best example is a person telneting to another computer and then telneting from there to the outside world. With a application proxy server the process is automated. As you telnet to the outside world the client send you to the proxy first. The proxy then connects to the server you requested (the outside world) and returns the data to you. Because proxy servers are handling all the communications, they can log everything they (you) do. For HTTP (web) proxies this includes very URL they you see. For FTP proxies this includes every file you download. They can even filter out "inappropriate" words from the sites you visit or scan for viruses.

Application proxy servers can authenticate users. Before a connection to the outside is made, the server can ask the user to login first. To a web user this would make every site look like it required a login.

SOCKS Proxy

A SOCKS server is a lot like an old switch board. It simply cross wires your connection through the system to another outside connection.

Most SOCKS server only work with TCP type connections. And like filtering firewalls they don't provide for user authentication. They can however record where each user connected to.

Choke Point:

A *choke point* forces attackers to use a narrow channel, which you can monitor and control. There are probably many examples of choke points in your life: the toll booth on a bridge, the check-out line at the supermarket, the ticket booth at a movie theatre.In network security, the firewall between your site and the Internet (assuming that it's the only connection between your site and the Internet) is such a choke point; anyone who's going to attack your site from the Internet is going to have to come through that channel, which should be defended against such attacks. You should be watching carefully for such attacks and be prepared to respond if you see them.

DMZ (demilitarized zone):

In computer networks, a DMZ (demilitarized zone) is a physical or logical sub-network that separates an internal local area network (LAN) from other untrusted networks, usually the Internet. External-facing servers, resources and services are located in the DMZ so they are accessible from the Internet but the rest of the internal LAN remains unreachable. This provides an additional layer of security to the LAN as it restricts the ability of hackers to directly access internal servers and data via the Internet.

Logging and alarms:

Logging is usually implemented at every device in the firewall, but these individual logs combine to become the entire record of user activity. logging devices will probably capture all hacker activities, including all user activities as well. The user can then tell exactly what a hacker is doing, and have such information available for audit. The audit log is an essential tool for detecting and terminating intruder attacks.

Many firewalls allow the user to preconfigure responses to unacceptable activities. The firewall should alert the user by several means. The two most common actions are for the firewall to break the TCP/IP connection, or to have it automatically set off alarms.

VPN: A VPN is an example of providing a controlled connectivity over a public network such as the Internet. VPNs utilize a concept called an IP tunnel—a virtual point-to-point link between a pair of nodes that are actually separated by an arbitrary number of networks. The virtual link is created within the router at the entrance to the tunnel by providing it with the IP address of the router at the far end of the tunnel. Whenever the router at the entrance of the tunnel wants to send a packet over this virtual link, it encapsulates the packet inside an IP datagram. The destination address in the IP header is the address of the router at the far end of the tunnel, while the source address is that of the encapsulating router.

Many consumers who use VPNs tend to also use hardware firewalls to further increase the security of their Internet connection. There are a few ways to combine the use of VPNs and firewalls. These ways differ in the way the location of the VPN, firewall, remote network, and the Internet are arranged in the connection. The most common three arrangements are as follows:

- Individual Computer to Remote Network via VPN through Firewall to Internet
- Individual Computer to Remote Network via Firewall to VPN to Internet
- Individual Computer to Remote Network via Firewall and VPN Combination in One Device to Internet

Types of Firewalls:

1)Packet Filters

2)Application-Level Gateways

3)Circuit-Level Gateways

1)Firewalls – Packet Filters:

-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 (from and to the internal network).

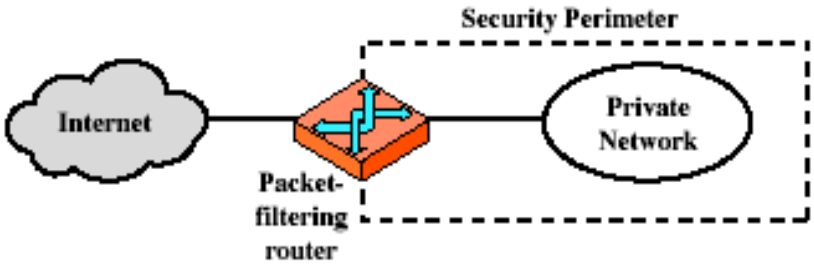
-Filtering rules are based on information contained in a network packet:

Source IP address: The IP address of the system that originated the IP packet (e.g., 192.168.1.1)

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

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

Default



(a) Packet-filtering router

-Packet the IP or TCP header.

-There a

i)**Defau** conservative. Initially, everything is blocked—services must be added on a case-by-case basis.

ii)**Default = forward:** that which is not expressly prohibited is permitted. It increases ease of use for end users but provides reduced security.

.

Table 20.1 Packet-Filtering Examples

A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

Attacks on Packet Filters

i)IP address spoofing attack-Here the Public network user will use the IP address of inside host(private network user) to send the packet.This is known as IP address spoofiing.

-In order to avoid it ,router can discard packets with 'inside host IP address' arriving on external interface.

ii)source routing attacks

-attacker sets a route other than default such that the routed packet will bypass the firewall. To

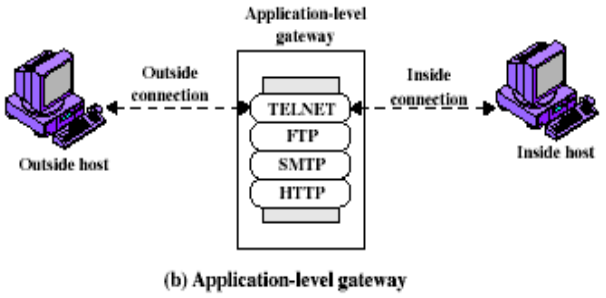
countermeasure this attack,block source routed packets

iii)tiny fragment attacks

-split header information over several tiny packets with a hope that one packet (first packet)will be checked by firewall .The sensitive information will be available in second,third packet and so on. This will not be checked by firewall,This is known as tiny fragment attack.

-To overcome this attack discard packets whose 'protocol type'=TCP and 'Header offset'=1.

2)Firewalls - Application Level Gateway (or Proxy)



(b) Application-level gateway

-Acts as relay of application-level traffic. The user contacts the gateway using a TCP/IP application, such as FTP, and the gateway asks the user for the name of a 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 points.

Note:

- application gateways , on the other hand, look up to Layer 7 packets of OSI model and have the understanding of protocols used by specific applications (e.g. it may not only make sure that TCP flags are legal for network mail software, but also make sure no illegal SMTP commands are sent over the connection).

Advantages:

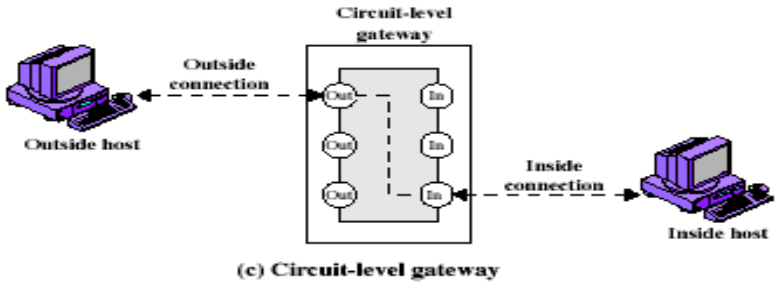
-Tend to be more secure than packet filters.

-It is easy to log and audit all incoming traffic at the application level.

Main Disadvantage

-Additional Processing overhead on each connection.

3)Firewalls - Circuit Level Gateway:



(c) Circuit-level gateway

-relays two TCP connections (one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host)

-imposes security by limiting which such connections are allowed

-once created usually relays traffic without examining contents

-typically used when trust internal users by allowing general outbound connections

-SOCKS (a protocol) commonly used for this.

Note:circuit based firewalls look Layer 3 packet of OSI model, and therefore enforce security for TCP/UDP protocls only (e.g. it can only check that say TCP packet flags are legal according to the TCP standard)

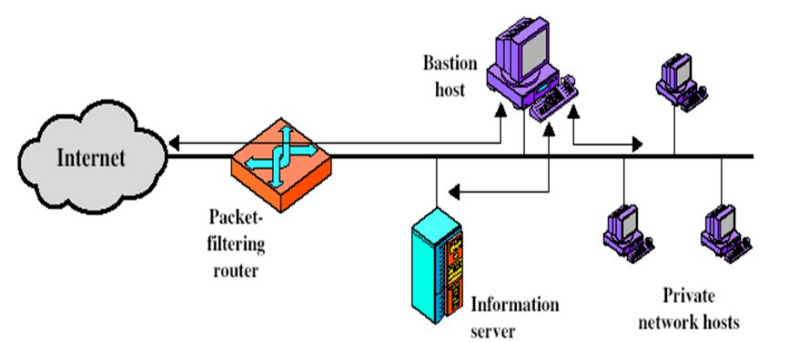
Bastion Host

-On the Internet, a bastion host is the only host computer that a company allows to be addressed directly from the public network and that is designed to screen the rest of its network from security exposure.

-highly secure host system that serves as a platform for an application-level or circuit-level gateway.

-only services that the network administrator considers essential are installed on the bastion host (e.g. Telnet, DNS, FTP, and user authentication)

Firewall Configurations



(a) Screened host firewall system (single-homed bastion host)

1)Single-Homed Bastion:

-In case of single homed bastion host the firewall system consists of a packet filtering router and a bastion host. A bastion host is basically a single computer with high security configuration, which has the following characteristics:

- Traffic from the Internet can only reach the bastion host; they cannot reach the internal network.
- Traffic having the IP address of the bastion host can only go to the Internet. No traffic from the internal network can go to the Internet.

-This type of configuration can have a web server placed in between the router and the bastion host in order to allow the public to access the server from the Internet. The main problem with the single homed bastion host is that if the packet filter route gets compromised then the entire network will be compromised. To eliminate this drawback we can use the dual homed bastion host firewall system.

2)Dual homed bastion host:

In this approach the bastion host is Dual Homed i.e. contains two NICs(Network Interface Card): one connected to the external network, and one connected to the internal network providing an additional layer of protection by requiring all traffic to go through the Bastion Host to move between the internal and external networks.

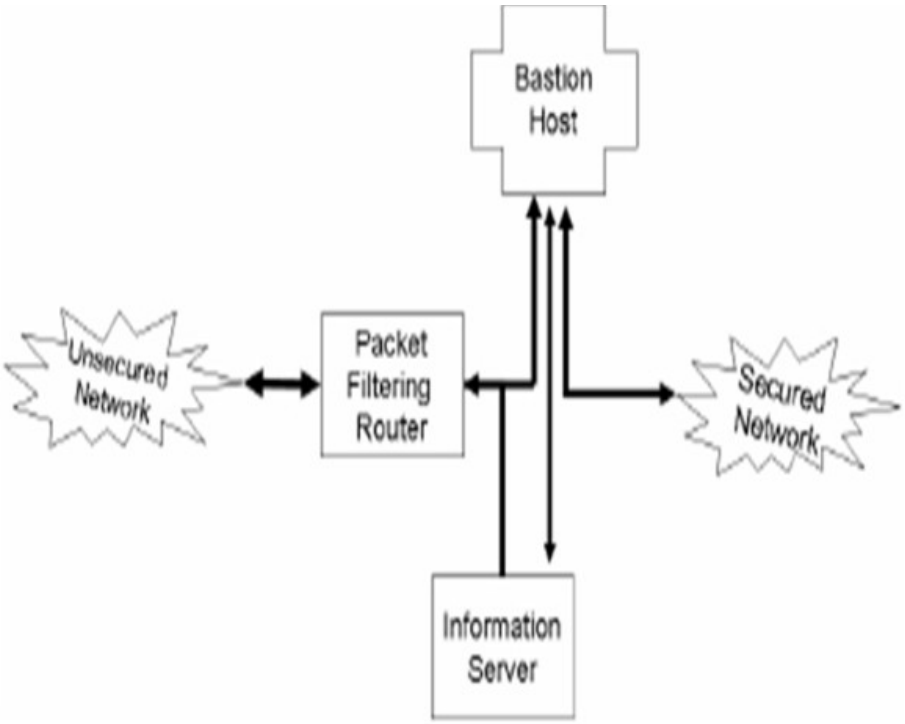


Fig: Dual-homed Firewall

3)Screened subnet firewalls:

A screened subnet (also known as a "triple-homed firewall") is a network architecture that uses a single firewall with three network interfaces.

- Interface 1 is the public interface and connects to the Internet.
- Interface 2 connects to a DMZ (demilitarized zone) to which hosted public services are attached.
- Interface 3 connects to an intranet for access to and from internal networks.

The purpose of the screened subnet architecture is to isolate the DMZ and its publicly-accessible resources from the intranet, thereby focusing external attention and any possible attack on that subnet. The architecture also separates the intranet and DMZ networks, making it more difficult to attack the intranet itself. When a properly configured firewall is combined with the use of private IP addresses on one or both of these subnets, attack becomes that much more difficult.

The screened subnet firewall configuration is the most secured environment. It has two packet-filtering routers, one between the bastion host and the Internet and the other between the bastion host and the internal network, creating an isolated sub-net. For all incoming traffic, the outside router (the router facing the internet) directs all the packets to the bastion host. The bastion host together with the firewall does preliminary filtering and if the packet passes the test, directs the packet to the less secure or DMZ, the demilitarized zone.

However, where the packet has to travel into the secure network, which is configured as a separate segment, the inside router along with the second firewall provides a second line of defence, managing DMZ access to the private network by accepting only traffic originating from the bastion host. For

outgoing traffic, the inside router manages private network access to the DMZ network. It permits internal systems to access only the bastion host. The filtering rules on the outside router require use of the proxy services by accepting outgoing traffic only from the bastion host.

It would be observed that in a screened sub-net firewall, the internal network (the private segment) is highly secured. The external firewall provides a measure of access control and protection for the DMZ systems consistent with their need for external connectivity.

The advantages of a screened subnet firewall system are explained here.

An intruder must penetrate three separate devices - the outside router, the bastion host, and the inside router to infiltrate the private network.

- Since the outside router advertises the DMZ network only to the Internet, systems on the Internet do not have routes to the protected private network. This ensures that the private network is invisible.
- Since the inside router advertises the DMZ network only to the private network, its systems do not have direct routes to the Internet. This ensures that inside users access the Internet via the proxy services residing on the bastion host.
- Packet-filtering routers direct traffic to specific systems on the DMZ network, eliminating the need for the bastion host to be dual-homed.
- Since the DMZ network is a different network than the private network, a Network Address Translator (NAT) can be installed on the bastion host to eliminate the need to renumber or re-subnet the private network.

INTRUDER

Definition:

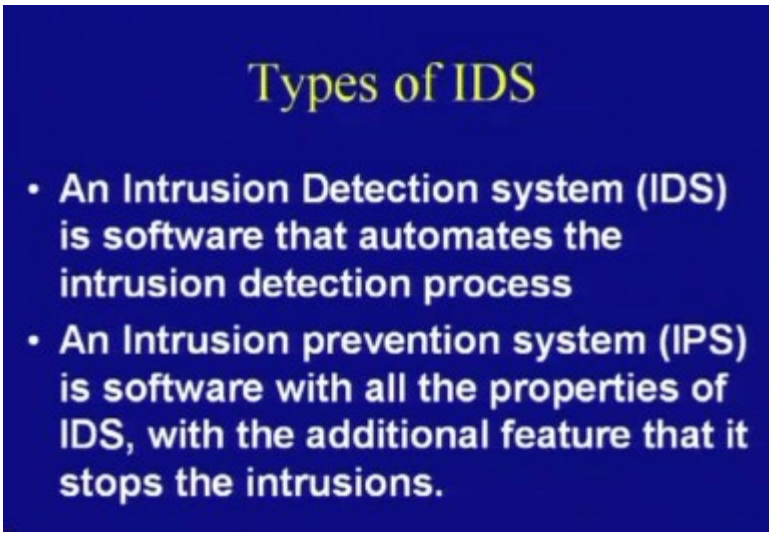
-An Intruder is a person who attempts to gain unauthorized access to a system, to damage that system, or to disturb data on that system. In summary, this person attempts to violate Security by interfering with system Availability, data Integrity or data Confidentiality.

Type of intruders:

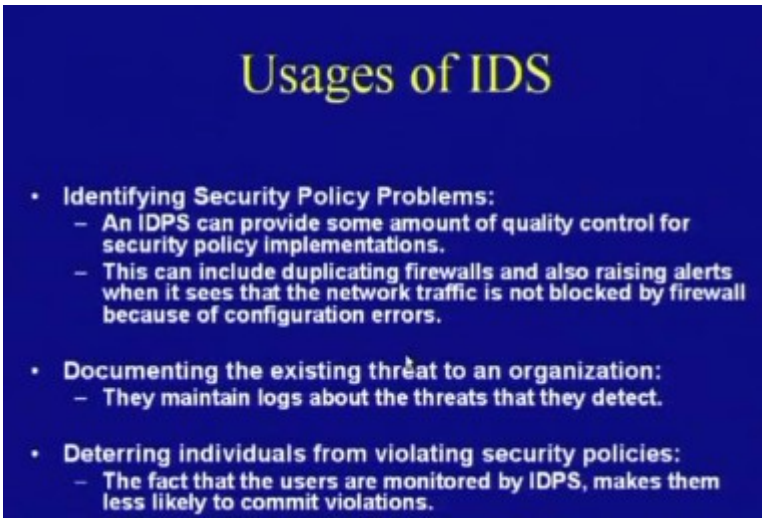
Types of Intruders

- **Masquerador:** They are typically outsiders from the trusted users and are not authorized to use the computer systems. These intruders penetrate the system protection by way of legitimate user accounts.
- **Misfeasor:** They are typically insiders and legitimate users who accesses resources that they are not authorized to use. Or, they may be authorized but misuses privileges.
- **Clandestine users:** They can be both insiders and outsiders. These type of intruders gain supervisory access to the system.

- **Few anti-intrusion techniques:**
  - **Prevention** – e.g. don't connect to the internet
  - **Preemption** – strike against threat before attack is mounted
  - **Deterrence** – increase perceived risk of negative consequences for the attacker
  - **Deflection** – e.g. use of honeypots
  - **Detection** – detect anomalies and notify authority to initiate proper response
  - **Countermeasures** – actively and autonomously counter intrusions as they are attempted



Usage of IDS:



-The normal uses of IDS's are essentially in monitoring the network and look for vulnerabilities, and also on the other hand to take some actions to prevent those attacks. But along with it we also have got some other functionalities like , they do some amount of quality control in which they actually not only continuously check that whether your security policies which are in place are correct or not. So they may take some steps, like if there are some incorrect settings to your firewalls they will do those corrections. They will raise alerts, whenever it sees that the network traffic is actually not locking some traffic because of configuration errors. So, basically it is kind of dials or quality control of the security policies which enforced the implementation and

security policies, and along with it very important functionality is to document the existing threat to an organization that is maintain logs about the threat that they detect. So later on, if there be some attacks people can revert back refer to these logs and can actually see what gone into the network.

- The other important functionality is like since IDPS are placed it generally deters individuals from violating security policies. So it is something like as we know that in shops if there are some CCT cameras , then the thieves are generally they essentially probably as more scared to do some criminal activities . So it is something like if we know that we continuously being monitored, the users know that they are be monitored byIDPS's makes them less likely to commit violations.

Base rate fallacy:

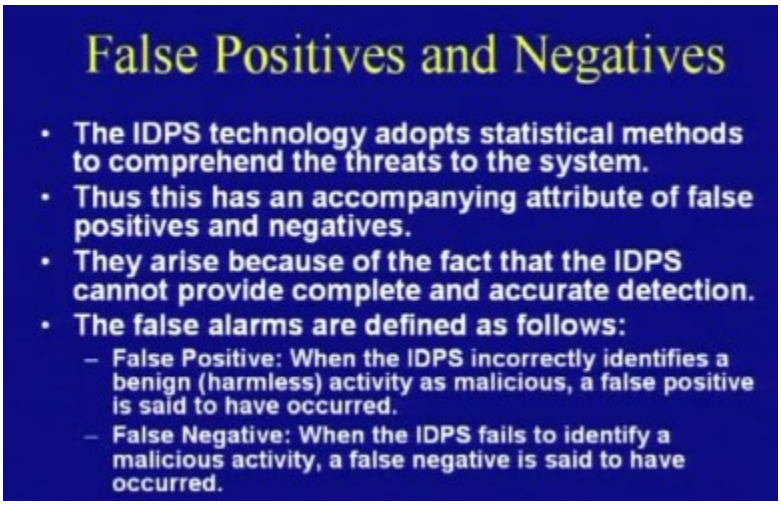
-IDS should avoid base rate fallacy problem.

-So what is base rate fallacy?

Definition: Ignoring statistical information in favor of using irrelevant information(that one incorrectly believes to be relevant) to make a judgment.

-A base rate fallacy is committed when a person judges that an outcome will occur without considering prior knowledge of the probability that it will occur. They focus on other information that isn't relevant instead. Imagine that I show you a bag of 250 M&Ms with equal numbers of 5 different colors. Then, I ask you what the probability is I will pick a green one while my eyes are closed? I also tell you that green M&Ms are my favorite and yesterday I picked out twice as many green M&Ms than red ones. If you ignored the fact that there are 50 of each color, and instead focused on the fact that I picked out twice as many green M&Ms than red yesterday, you have commited a base rate fallacy because what I did yesterday is irrelevant information.

-Base rate fallacy will result in false positive and false negatives.So IDS has to avoid base rate fallacy problem.



Audit record analysis(Tools)

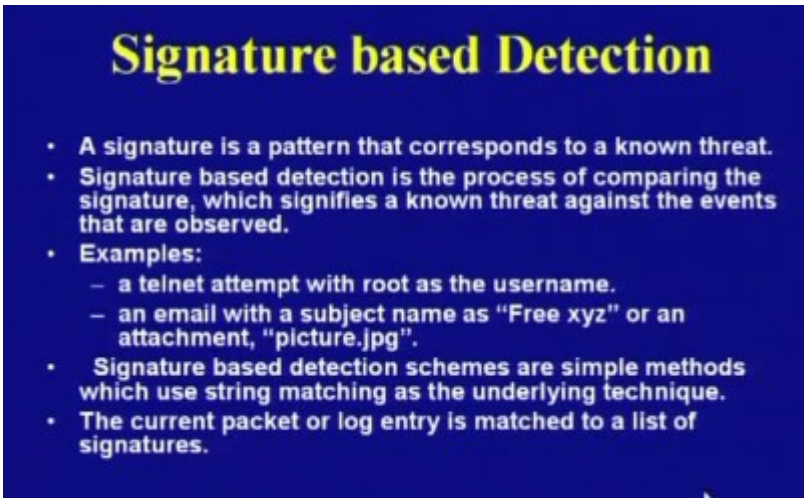
- use various tests on these to determine if current behavior is acceptable
  - mean & standard deviation of a parameter over some period.
  - Multivariate(correlation between 2 or more variables)
  - markov process(to show transition between 2 states)
  - time series(to know sequences of events that happen too slowly or fastly)
  - operational(to detect any abnormal behavior like exceeding some limit)

Classification of Intrusion Detection Systems:

a)Signature-Based IDS :

A signature based IDS monitor's packets in the network and compares with preconfigured and predetermined attack patterns known as signatures.

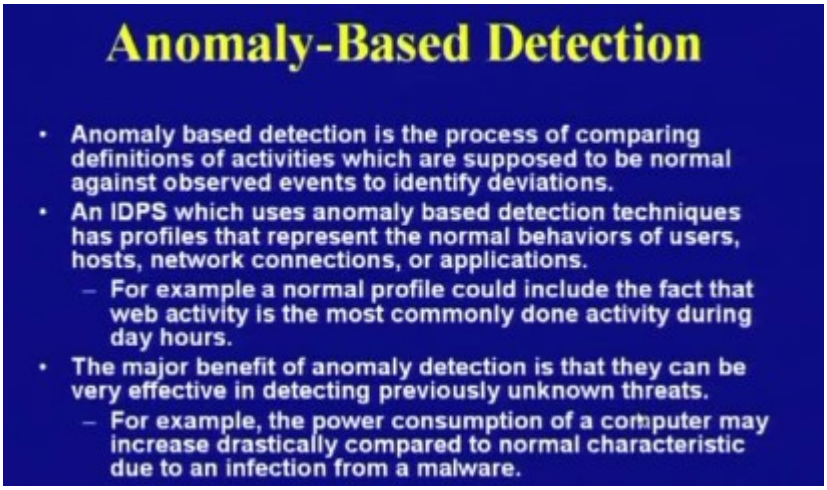
When a new attack is recognized experts or programs have to identify typical patterns in such attacks, which can be made into signature. Since this process takes time, there will be a lag between the new threat discovered and signature being applied in IDS for detecting the threat. During this lag time your IDS will be unable to identify the threat. To reduce further lag, security software using such signatures should be updated as frequently as feasible.



b)Anomaly-based detection:

-It records what sort of bandwidth is generally used, what kind of protocols are used, which ports and devices generally connect to each other- and alert the administrator or user when

traffic is detected which is anomalous (not normal).



Types of Anomaly-based detection:

1)Statistical based anomaly detection

2)Profile based anomaly detection

c)Host Based Intrusion Detection System:

-Host based intrusion detection (HIDS) refers to intrusion detection that takes place on a single host system. The data is collected from an individual host system. The HIDS agent monitors activities such as integrity of system, application action, file changes, host based network traffic, and system logs.

- By using common hashing tools, file timestamps, system logs, and monitors system calls and the local network interface gives the agent insight to the present state of the local host. If there is any unauthorized change or activity is detected, it alerts the user by a pop-up, it alerts the central management server, blocks the activity, or a combination of the above three. The decision should be based on the policy that is installed on the local system.

d)Rule-based IDS:

– Involves attempt to define a set of rules that can be used to decide whether a users behaviour is that of an intruders

a) Anomaly detection:

– Rules are developed to detect deviation from previous usage patterns

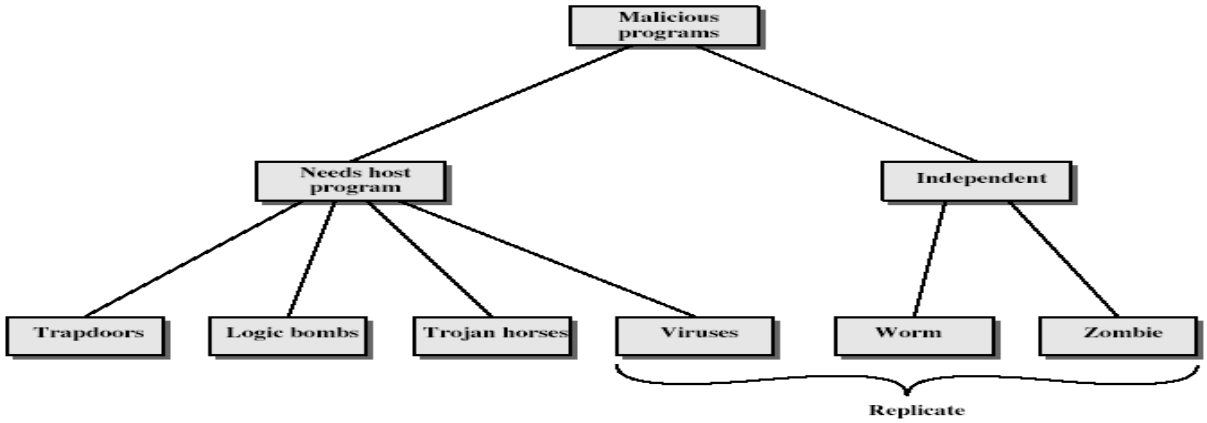
b) Penetration Identification:

– An expert systems approach that searches for suspicious behaviour

Viruses:

Malicious code is the term used to describe any code in any part of a software system or script that is intended to cause undesired effects, security breaches or damage to a system. Malicious code describes a broad category of system security terms that includes attack scripts, viruses, worms, Trojan horses, backdoors, and malicious active content

Malicious Software:



A **trapdoor or backdoor** is a feature in a program by which someone can access the program

other than by the obvious, direct call, perhaps with special privileges. For instance, an automated bank teller program might allow anyone entering the number 990099 on the keypad to process the log of everyone's transactions at that machine. In this example, the trapdoor could be intentional, for maintenance purposes, or it could be an criminal way for the implementer to wipe out any record of a crime.

Logic bombs may reside within standalone programs, or they may be part of worms or viruses. An example of a logic bomb would be a virus that waits to execute until it has infected a certain number of hosts. A **time bomb** is a subset of logic bomb, which is set to trigger on a particular date and/or time. An example of a time bomb is the infamous 'Friday the 13th' virus.

Trojan Horses: A Trojan is a malware that performs unauthorized, often malicious, actions. The main difference between a Trojan and a virus is the inability to replicate itself. Like a virus, a Trojan can cause damage or an unexpected system behavior, and can compromise the security of the visited systems; but, unlike viruses, it does not replicate. A Trojan looks like any normal program, but it has some hidden malicious code within it.

A computer **virus i**s a self replicating computer program which can attach itself to other files/programs, and can execute secretly when the host program/file is activated. When the virus is executed, it can perform a number of tasks, such as erasing your files/hard disk, displaying nuisance information, attaching to other files, etc.

A **virus can be either transient or resident.** A **transient virus** has a life that depends on the life of its host; the virus runs when its attached program executes and terminates when its attached program ends. (During its execution, the transient virus may have spread its infection to other programs.) A **resident virus** locates itself in memory; then it can remain active or be activated as a stand-alone program, even after its attached program ends.

A **worm** is a program that spreads copies of itself through a network. The primary difference between a worm and a virus is that a worm operates through networks, and a virus can spread through any medium (but usually uses copied program or data files). Additionally, the worm spreads copies of itself as a stand-alone program, whereas the virus spreads copies of itself as a program that attaches to or embeds in other programs.

White et al. also define a **rabbit** as a virus or worm that self-replicates without bound, with the intention of exhausting some computing resource. A rabbit might create copies of itself and store them on disk, in an effort to completely fill the disk, for example.

TABLE 3-1 Types of Malicious Code.

Code Type	Characteristics
Virus	Attaches itself to program and propagates copies of itself to other programs
Trojan horse	Contains unexpected, additional functionality
Logic bomb	Triggers action when condition occurs
Time bomb	Triggers action when specified time occurs
Trapdoor	Allows unauthorized access to functionality
Worm	Propagates copies of itself through a network
Rabbit	Replicates itself without limit to exhaust resource

Type of virus

Memory-Resident Virus

This type will reside in main system memory. Whenever the operating system executes a file, the virus will infect a file if it is a suitable target, for example, a program file.

Program File Virus

This will infect programs like EXE, COM, SYS etc.

Polymorphic Virus

The virus itself can change form using various polymorphism techniques.

Boot Sector Virus

This type will infect the system area of a disk, when the disk is accessed initially or booted.

Stealth Virus

A virus which uses various stealth techniques in order to hide itself from detection by anti-virus software.

Macro Virus

Unlike other virus types, these viruses attack data files instead of executable files.

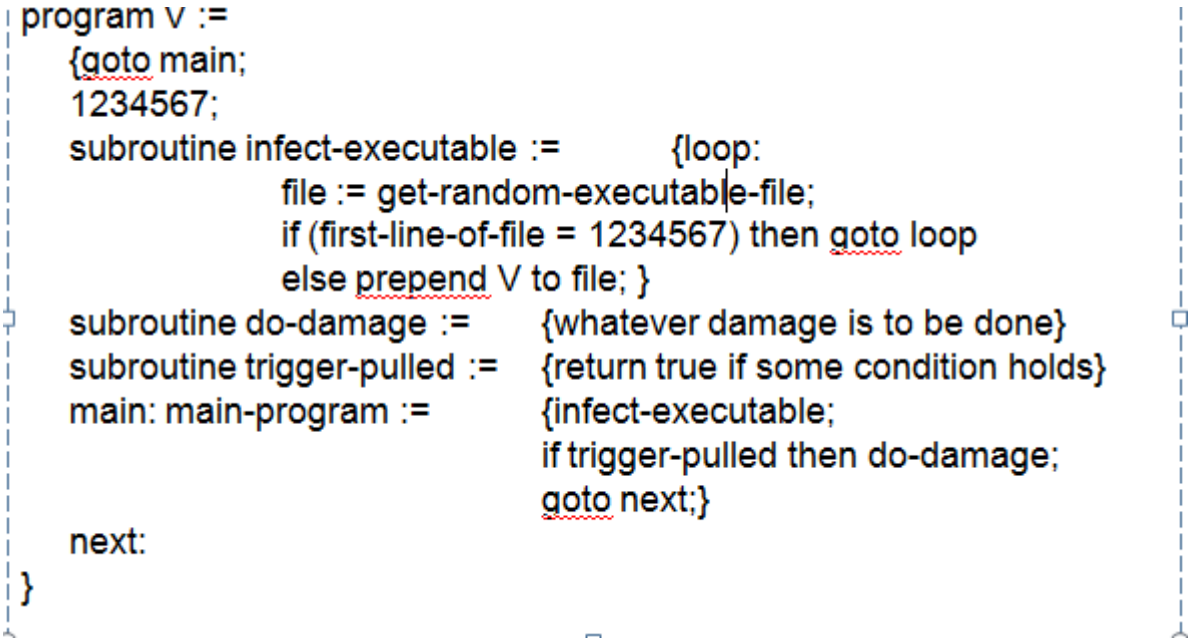
Macro viruses are particularly common due to the fact that:

- They attach to documents and files, which are platform independent.
- The document is sent to other computers by, for example, email or file exchange. Recipients are receiving the infected document from a "trusted" sender.

Email virus

A virus spread by email messages.

Virus structure:



Virus Countermeasures

- viral attacks exploit lack of integrity control on systems
- to defend need to add such controls
- typically by one or more of:
  - **prevention** - block virus infection mechanism
  - **detection** - of viruses in infected system
  - **reaction** - restoring system to clean state

Advanced Anti-Virus Techniques

- generic decryption
  - use CPU simulator to check program signature & behavior before actually running it
- digital immune system (IBM)
  - general purpose emulation & virus detection
  - any virus entering org is captured, analyzed, detection/shielding created for it, removed

Behavior-Blocking Software

- integrated with host O/S
- monitors program behavior in real-time
  - eg file access, disk format, executable mods, system settings changes, network access
- for possibly malicious actions
  - if detected can block, terminate, or seek ok
- has advantage over scanners
- but malicious code runs before detection

Trusted Systems

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology. This section provides a brief overview of this topic. We begin by looking at some basic concepts of data access control.

Before delving into the architecture of a trusted system it is necessary to define the context in which we use “trust.” In the context of trusted systems this term entails that the legitimate owner of the information believes that the information is being used appropriately.

- For example if I require that jetBlue be able to use my personal information but no other entity will have access to my information, then a trusted system would make sure that this belief is accurate (by restricting all other uses other than those that I specify). -As another example if a company asks for my delivery address to be used to deliver goods during the next week, I could specify that I want my address destroyed after a week and in a “trusted” system this would happen.