Cryptography & Network Security (20A05702b)

LECTURE NOTES

IV-B.TECH & I SEM

VEMU INSTITUTE OF TECHNOLOGY

(Approved By AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu) Accredited By NAAC, NBA (EEE, ECE & CSE) & ISO: 9001-2015 Certified InstitutionNear Pakala, P.Kothakota, Chittoor- Tirupathi Highway Chittoor, Andhra Pradesh-517 112Web Site: www.vemu.org

UNIT-1

1. Introduction:

Network security: Introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. Network security measures are needed to protect data during their transmission.

SECURITY CONCEPTS

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/ data, and telecommunications).

This definition introduces three key objectives that are at the heart of computer security.

1) **Confidentiality:** The data must be hidden from unauthorized access:

Data confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

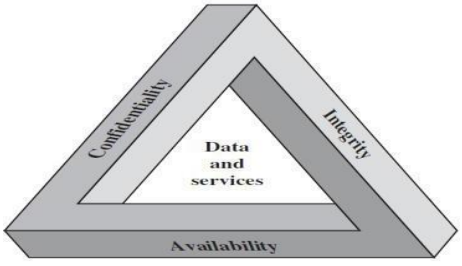
2) **Integrity:** The data can be protected from unauthorized change:

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

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

3) Availability: The data may be available to the authorized users when it is needed.

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



OSI SECURITY ARCHITECTURE

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

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

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

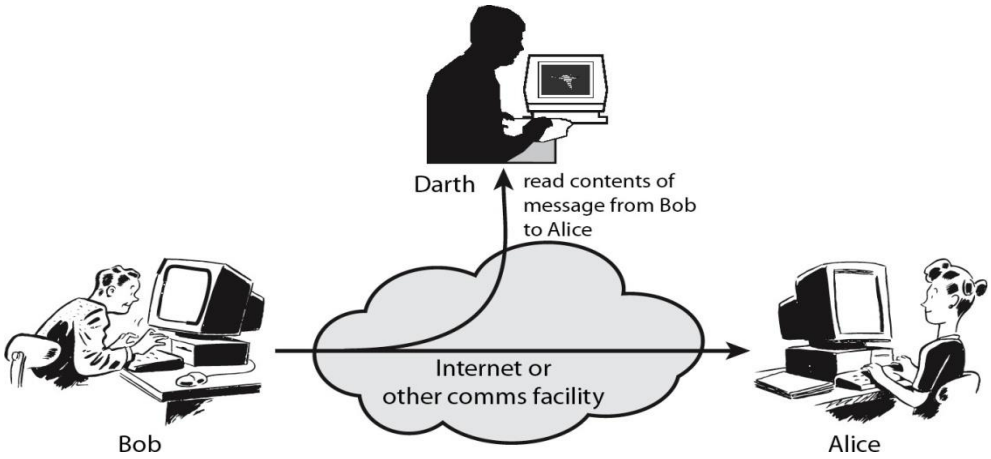
SECURITY ATTACKS

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

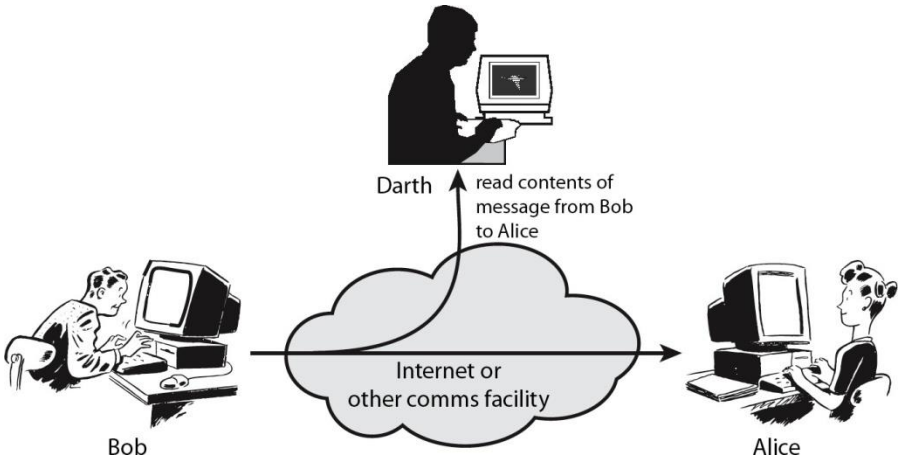
A useful means of classifying security attacks is in terms of passive attacks and active attacks. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

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

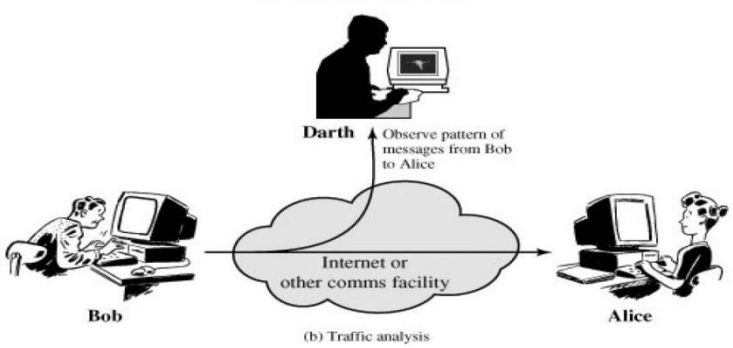


The release of message contents is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.



Release of Message

A second type of passive attack, traffic analysis. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption.



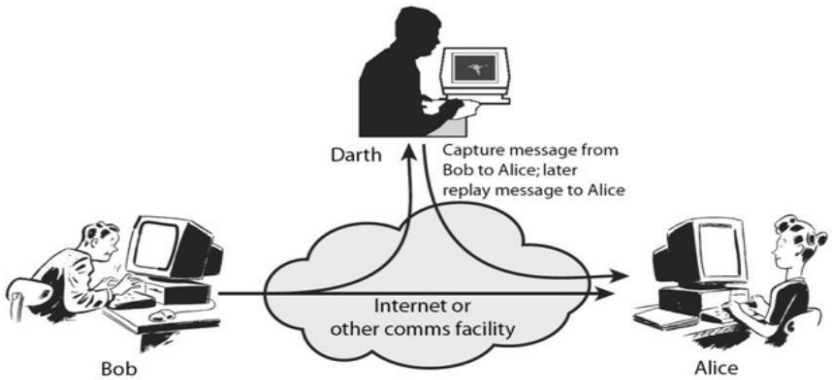
Traffic analysis

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

Active Attacks

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

A masquerade takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack.



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

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message meaning “Allow John Smith to read confidential file accounts” is modified to mean “Allow Fred Brown to read confidential file accounts.”

(c) Modification of messages

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

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All of the techniques for providing security have two components:

A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.

Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

SECURITY SERVICES

It is a processing or communication service that is provided by a system to give a specific kind of production to system resources. Security services implement security policies and are implemented by security mechanisms.

Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. It is used to prevent the disclosure of information to unauthorized individuals or systems. It has been defined as "ensuring that information is accessible only to those authorized to have access".

The other aspect of confidentiality is the protection of traffic flow from analysis.

Ex: A credit card number has to be secured during online transaction.

Authentication

This service assures that a communication is authentic. For a single message transmission, its function is to assure the recipient that the message is from intended source. For an ongoing interaction two aspects are involved. First, during connection initiation the service assures the authenticity of both parties. Second, the connection between the two hosts is not interfered allowing a third party to masquerade as one of the two parties. Two specific authentication services defines in X.800 are

Peer entity authentication: Verifies the identities of the peer entities involved in communication. Provides use at time of Media connection establishment and during data transmission. Provides confidence against a masquerade or replay attack

Data origin authentication: Assumes the authenticity of source of data unit, but does not provide protection against duplication or modification of data units. Supports applications like electronic mail, where no prior interactions take place between communicating entities.

Integrity

Integrity means that data cannot be modified without authorization. Like confidentiality, it can be applied to a stream of messages, a single message or selected fields within a message. Two types of integrity services are available. They are:

Connection-Oriented Integrity Service: This service deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering or replays. Destruction of data is also covered here. Hence, it attends to both message stream modification and denial of service.

Connectionless-Oriented Integrity Service: It deals with individual messages regardless of larger context, providing protection against message modification only. An integrity service can be applied with or without recovery. Because it is related to active attacks, major concern will be detection rather than prevention. If a violation is detected and the service reports it, either human intervention or automated recovery machines are required to recover.

Non-repudiation

Non-repudiation prevents either sender or receiver from denying a transmitted message. This capability is crucial to e-commerce. Without it an individual or entity can deny that he, she or it is responsible for a transaction, therefore not financially liable.

Access Control

This refers to the ability to control the level of access that individuals or entities have to a network or system and how much information they can receive. It is the ability to limit and control the access to host systems and applications via communication links. For thiseach entity trying to gain access must first be identified or authenticated, so that access rights can be tailored to the individuals.

Availability

It is defined to be the property of a system Media or a system resource being accessible and usable upon demand by an authorized system entity. The vilability can significantly be affected by a variety of attacks, some amenable to automated counter measures i.e authentication and encryption and others need some sort of physical action to prevent or recover from loss of availability of elements of distributed system.

SECURITY MECHANISMS

According to X.800, the security mechanisms are divided into those implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. X.800 also differentiates reversible & irreversible encipherment mechanisms.

Specific Security Mechanisms

Incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment: It refers to the process of applying mathematical algorithms for converting data into a form that is not intelligible. This depends on algorithm used and encryption keys.

Digital Signature: The appended data or a cryptographic transformation applied to any data unit allowing to prove the source and integrity of the data unit and protect against forgery.

Access Control: A variety of techniques used for enforcing access permissions to the system resources.

Data Integrity: A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

Authentication Exchange: A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding: The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control: Enables selection of particular physically secure routes for certain data and allows routing changes once a breach of security is suspected.

Notarization: The use of a trusted third party to assure cert in properties of a data exchange

Pervasive Security Mechanisms

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

Trusted Functionality: That which is perceived to b correct with respect to some criteria

Security Level: The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection: It is the process of detecting all the events related to network security.

Security Audit Trail: Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery: It deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

A MODEL FOR NETWORK SECURITY:

A Network Security Model exhibits how the security service has been designed over the network to prevent the opponent from causing a threat to the confidentiality or authenticity of the information that is being transmitted through the network.

For a message to be sent or receive there must be a sender and a receiver. Both the sender and receiver must also be mutually agreeing to the sharing of the message. Now, the transmission of a message from sender to receiver needs a medium i.e. Information channel which is an Internet service.

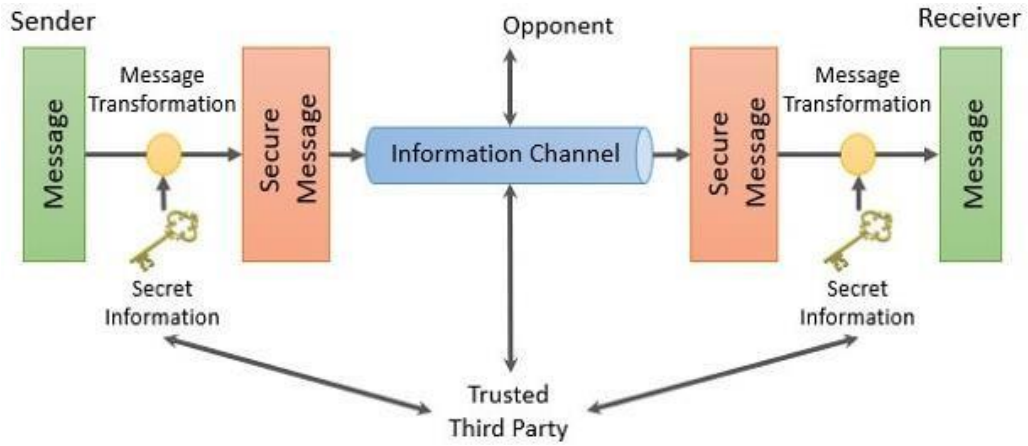
A logical route is defined through the network (Internet), from sender to the receiver and using the communication protocols both the sender and the receiver established communication.

Any security service would have the three components discussed below:

Transformation of the information which has to be sent to the receiver. So, that any opponent present at the information channel is unable to read the message.

Sharing of the secret information between sender and receiver of which the opponent must not any clue. Yes, we are talking of the encryption key which is used during the encryption of the message at the sender's end and also during the decryption of message at receiver's end.

There must be a trusted third party which should take the responsibility of distributing the secret information (key) to both the communicating parties and also prevent it from any opponent.



A Model for Network Security

The network security model presents the two communicating parties sender and receiver who mutually agrees to exchange the information. The sender has information to share with the receiver.

So, considering this general model of network security, one must consider the following four tasks while designing the security model.

1. To transform a readable message at the sender side into an unreadable format, an appropriate algorithm should be designed such that it should be difficult for an opponent to crack that security algorithm.

2. Next, the network security model designer is concerned about the generation of the secret information which is known as a key.

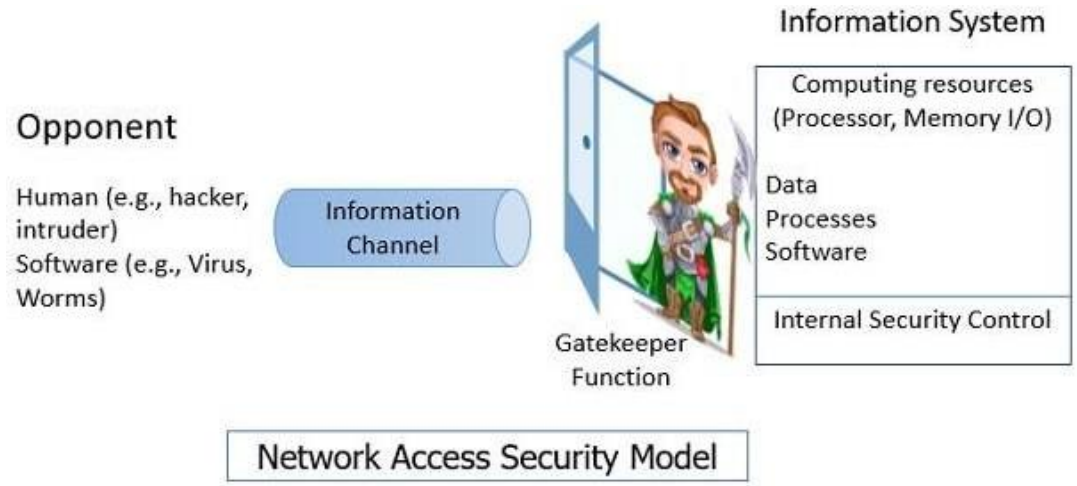
This secret information is used in conjunction with the security algorithm in order to transform the message.

3. Now, the secret information is required at the ends, sender's end and receiver's end. At sender's end, it is used to encrypt or transform the message into unreadable form and at the receiver's end, it is used to decrypt or retransform the message into readable form. So, there must be a trusted third party who will distribute the secret information to both sender and receiver. While designing the network security model designer must also concentrate on developing the methods to distribute the key to the sender and receiver. An appropriate methodology must be used to deliver the secret information to the communicating parties without the interference of the opponent.

NETWORK ACCESS SECURITY MODEL

Network access security model which is designed to secure the information system which can be accessed by the attacker through the network.
Attackers who attack your system that is accessible through the internet. These attackers fall into two categories:
1. Hacker: The one who is only interested in penetrating into your system. They do not cause any harm to your system they only get satisfied by getting access to your system.
2. Intruders: These attackers intend to do damage to your system or try to obtain the information from the system which can be used to attain financial gain.

The attacker can place a logical program on your system through the network which can affect the software on your system. This leads to two kinds of risks:
a. Information threat: This kind of threats modifies data on the user's behalf to which actually user should not access. Like enabling some crucial permission in the system.
b. Service threat: This kind of threat disables the user from accessing data on the system.



There are two ways to secure your system from attacker of which the first is to introduce the gatekeeper function. Introducing gatekeeper function means introducing login-id and passwords which would keep away the unwanted access.
In case the unwanted user gets access to the system the second way to secure your system is introducing internal control which would detect the unwanted user trying to access the system by analyzing system activities. This second method we call as antivirus which we install on our system to prevent the unwanted user from accessing your computer system through the internet.

TERMINOLOGIES OF CRYPTOGRAPHY:

Cryptography

The many schemes used for encryption constitute the area of study known as cryptography

Crypt analysis

Techniques used for Decode a message without any knowledge of the enciphering details fall into the area of cryptanalysis. Cryptanalysis is what the layperson calls "breaking the code."

Cryptology

The areas of cryptography and cryptanalysis together are called cryptology

Plain Text

This is the original intelligible message or data that is fed into the algorithm as input.

Cipher Text

This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts. The cipher text is an apparently random stream of data and, as it stands, is unintelligible.

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.

Encryption

The process of converting from plaintext to cipher text

Decryption

The process of restoring the plaintext from the cipher text

Enciphering Algorithm

The encryption algorithm performs various substitutions and transformations on the plaintext

Deciphering Algorithm

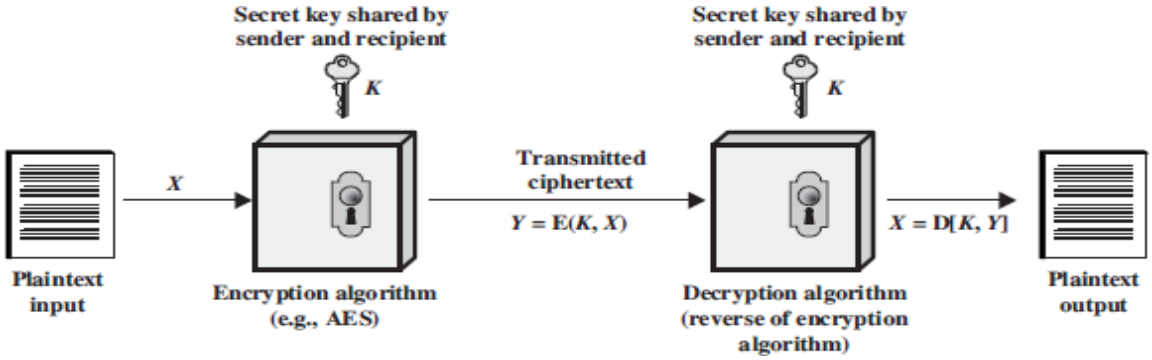
This is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plaintext.

SYMMETRIC CIPHER MODEL (OR) CONVENTIONAL ENCRYPTION MODEL

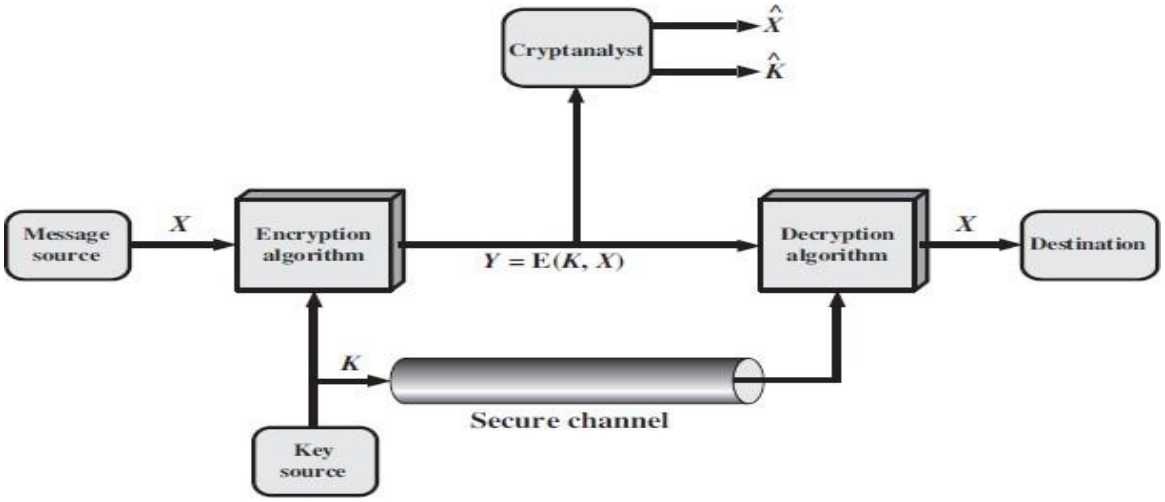
A symmetric encryption scheme has five ingredients. They are Plain Text, Encryption Algorithm, Secret Key, Decryption Algorithm, Cipher Text

There are two requirements for secure use of conventional encryption:

- We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more cipher texts would be unable to decipher the cipher text or figure out the key.
- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

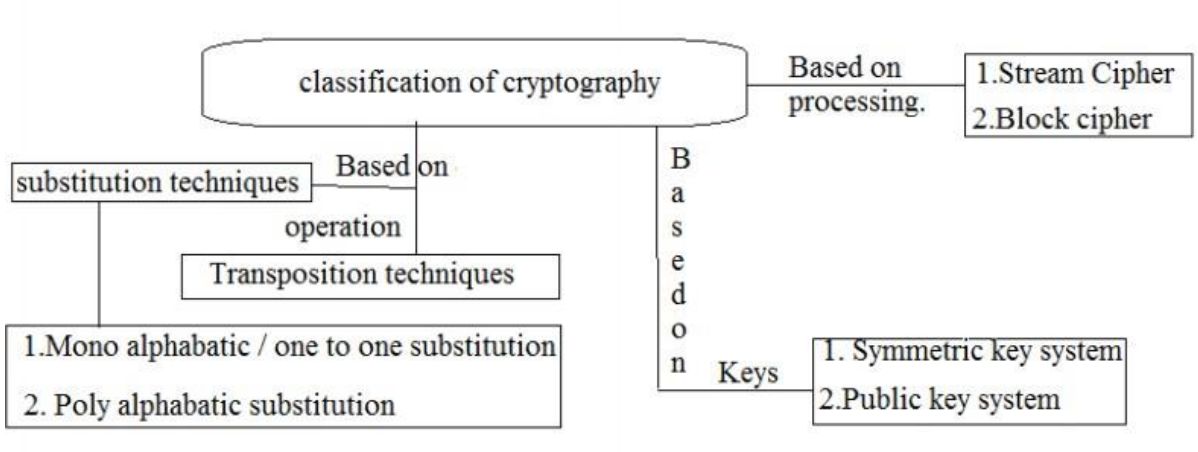


Model of Symmetric Encryption



Model of Symmetric Cryptosystem

CRYPTOGRAPHY CLASSIFICATIONS:



CLASSICAL ENCRYPTION TECHNIQUES

SUBSTITUTION CIPHER

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

CAESAR CIPHER

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

For example,

Plain text : meet me after the toga party
Cipher Text : PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

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

Let us assign a numerical equivalent to each letter:

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

Then the algorithm can be expressed as follows.
For each plaintext letter p, substitute the cipher text letter C
$C = E(3, p) = (p + 3) \bmod 26$

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

The decryption algorithm is simply
$p = D(k, C) = (C\ k) \bmod 26$
If it is known that a given cipher text is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys.

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

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

PLAY FAIR CIPHER

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

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

In this case, the keyword is **monarchy**.

The matrix is constructed by filling in the letters of the keyword (minus duplicates) from **left to right** and from **top to bottom**, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.

Plaintext is encrypted two letters at a time, according to the following rules:

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

HILL CIPHER

Another interesting multi-letter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. The encryption algorithm takes m successive plaintext letters and substitutes for them m cipher text letters.

The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1 ... z = 25).
For m = 3, the system can be described as follows:

$c_1 = (k_{11}P_1 + k_{12}P_2 + k_{13}P_3) \bmod 26$
$c_2 = (k_{21}P_1 + k_{22}P_2 + k_{23}P_3) \bmod 26$
$c_3 = (k_{31}P_1 + k_{32}P_2 + k_{33}P_3) \bmod 26$

C = PK mod 26
Where C and P are column vectors of length 3, representing the plaintext and cipher text, and K is a 3 x 3 matrix, representing the encryption key. Operations are performed in mod 26.

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

ONE TIME PAD

An Army Signal Corp officer, Joseph Mauborgne suggested using a **random key** that is as **long as the message**, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a one-time pad, is **unbreakable**.

It produces random output that bears no statistical relationship to the plaintext. Because the cipher text contains no information whatsoever about the plaintext, there is simply no way to break the code.

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

Consider the
cipher text : ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
We now show two different decryptions using two different keys:

key 1: pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plain text: mr mustard with the candlestick in the hall

key 2: mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt
plain text: miss scarlet with the knife in the library

If the actual key were produced in a truly random fashion, then the cryptanalyst cannot say that one of these two keys is more likely than the other. Thus, there is no way to decide which key is correct and therefore which plaintext is correct. Therefore, the code is unbreakable.

TRANSPOSITION CIPHER

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

RAIL FENCE TECHNIQUE

The simplest 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,
to encipher the message "**meet me after the toga party**" with a rail fence of depth 2, we write the following:

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

The encrypted message is **"MEMATRHTGPRYETEFETEOAAT"**

SIMPLE COLUMNAR TECHNIQUE

A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm.
For example,
Key: 3 4 2 1 5 6 7
Plaintext: **a t t a c k p o s t p o n e d u n t i l t w o a m x y z**

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

Cipher text: **TTNAAPTMTSUOAODWCOIXKNLYPETZ**

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext.

For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the cipher text in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed.
Thus, if the foregoing message is re-encrypted using the same algorithm,

Key: 3 4 2 1 5 6 7

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

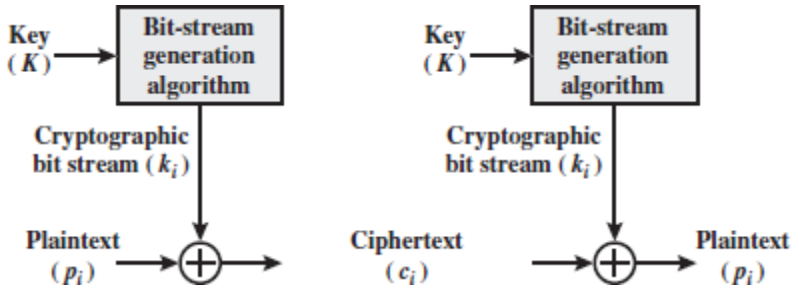
Cipher text: **NSCYAUOPTTWLTTDNPOIETAXTMOKZ**

STEGANOGRAPHY
A plaintext message may be hidden in any one of the two ways.
The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text. A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message. e.g., (i) the sequence of first letters of each word of the overall message spells out the real (hidden) message. (ii) Subset of the words of the overall message is used to convey the hidden message. Various other techniques have been used historically, some of them are:
 Character marking – selected letters of printed or typewritten text are overwritten in pencil
The marks are ordinarily not visible unless the paper is held to an angle to bright light.

 Invisible ink – a number of substances can be used for writing but leave no visible trace Until heat or some chemical is applied to the paper.
 Pin punctures – small pin punctures on selected letters are ordinarily not visible unless the Paper is held in front of the light.
 Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

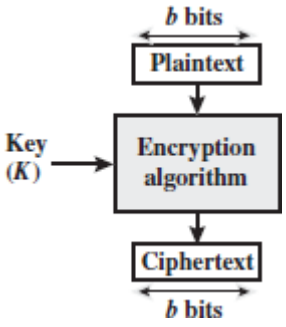
COMPARISON OF STREAM CIPHERS AND BLOCK CIPHERS

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the auto keyed Vigenère cipher and the Vernam cipher.
In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the key stream is as long as the plaintext bit stream. If the cryptographic key stream is random, then this cipher is unbreakable by any means other than acquiring the keystream. However, the key stream must be provided to both users in advance via some independent and secure channel. This introduces insurmountable logistical problems if the intended data traffic is very large.
Accordingly, for practical reasons, the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users. In this approach, the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong. Now, the two users need only share the generating key, and each can produce the key stream.



Stream Cipher using algorithmic bit-stream generator

A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically, a block size of 64 or 128 bits is used. In general, they seem applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers.



Block Cipher

FEISTEL BLOCK CIPHER

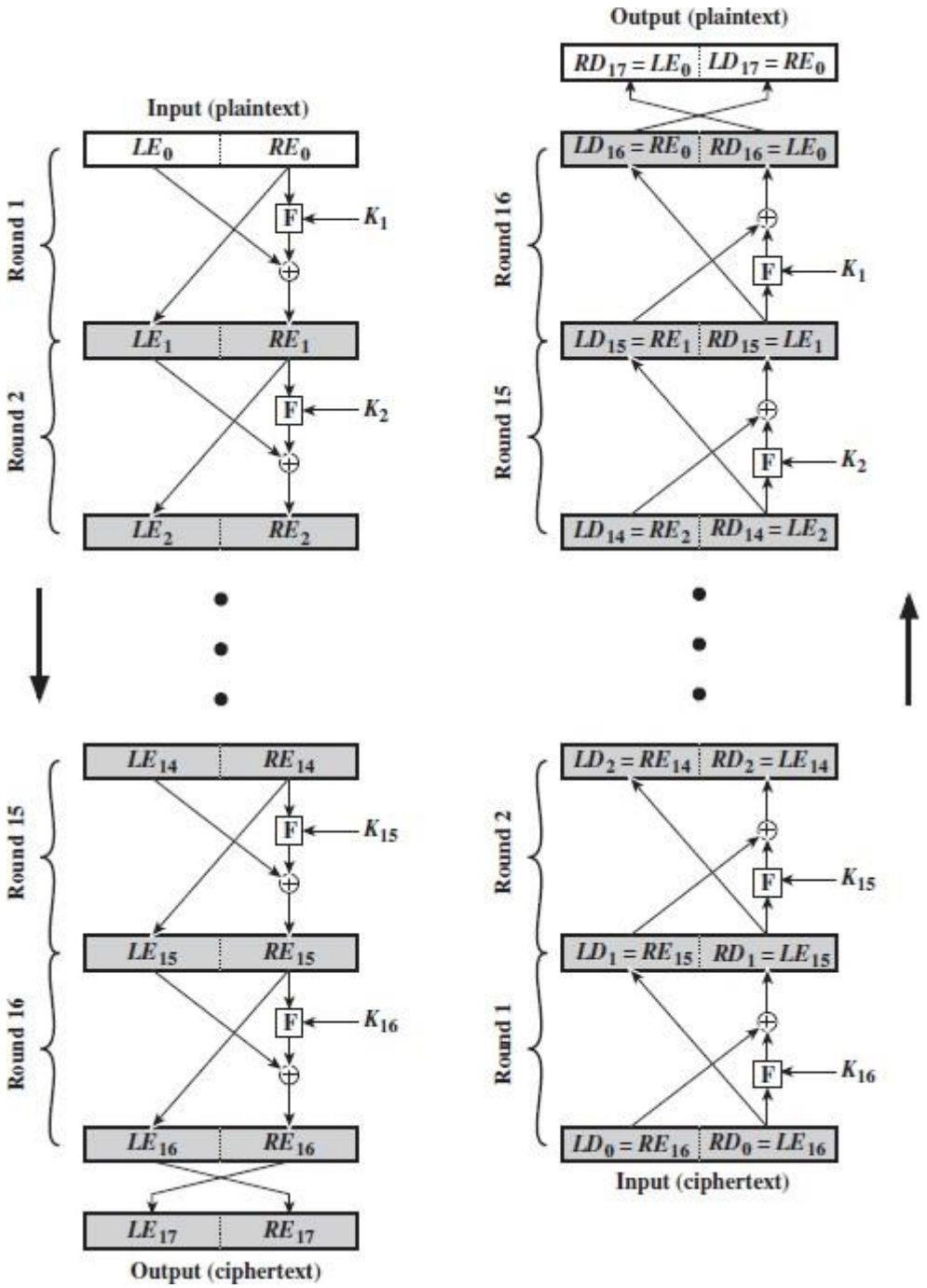
Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers The essence of the approach is to develop a block cipher with a key length of *k* bits and a block length of bits, allowing a total of 987possible transformations, rather than the! Transformations available with the ideal block cipher.
In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:
• **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
• **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

Diffusion: A cryptographic technique that seeks to obscure the statistical structure of the plaintext by spreading out the influence of each individual plaintext digit over many cipher text digits.

Confusion: A cryptographic technique that seeks to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible. This is achieved by the use of a complex scrambling algorithm that depends on the key and the input.

The inputs to the encryption algorithm are a plaintext block of length **2w** bits and a key. The plaintext block is divided into two halves L_0 and R_0. The two halves of the data pass through rounds of processing and then combine to produce the cipher text block. Each round i has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a sub key K_i derived from the overall K. In general, the sub keys K_i are different from K and from each other. 16 rounds are used, although any number of rounds could be implemented.

All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round sub key K_i . Another way to express this is to say that F is a function of right-half block of w bits and a sub key of y bits, which produces an output value of length *w* bits: $F(REi, Ki+1)$.
Followingthis substitution, a



Feistel Encryption and Decryption

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

• **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable trade off and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

• **Key size:** Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.

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

• **Sub key generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

• **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.

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

BLOCK CIPHER MODES OF OPERATION

When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise. To apply a block cipher in a variety of applications, five modes of operation have been defined by NIST. In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream.

(i) Electronic Code Book (ECB) Mode

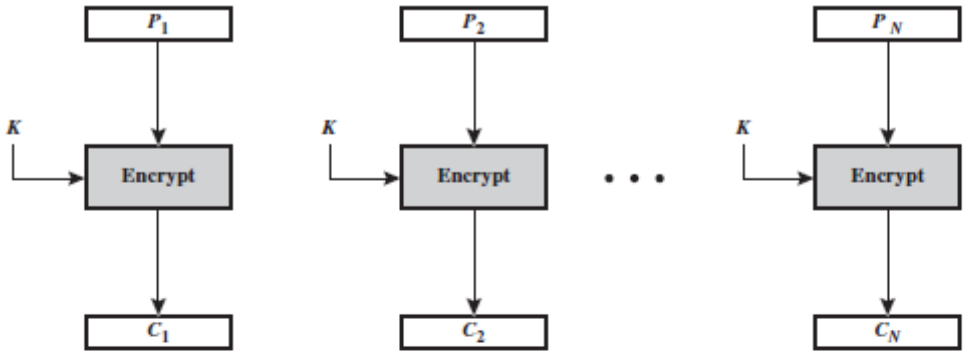
This mode is a most straightforward way of processing a series of sequentially listed message blocks.

Operation

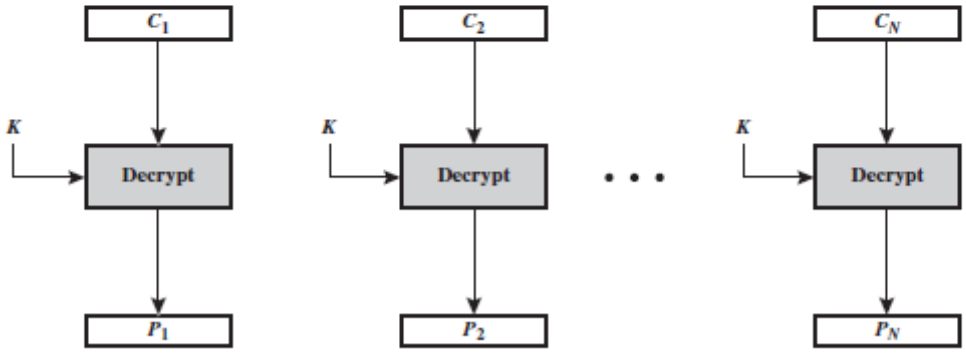
- The user takes the first block of plaintext and encrypts it with the key to produce the first block of cipher text.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

The ECB mode is **deterministic**, that is, if plaintext block P1, P2,…, Pm are encrypted twice under the same key, the output cipher text blocks will be the same.
In fact, for a given key technically we can create a codebook of cipher texts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding cipher text. Thus, the operation is analogous to the assignment of code words in a codebook, and hence gets an official name: Electronic Codebook mode of operation (ECB). It is illustrated as follows:

ECB	$C_j = E(K, P_j)$ $j = 1, \ldots, N$	$P_j = D(K, C_j)$ $j = 1, \ldots, N$



(a) Encryption



(b) Decryption

ECB - Encryption and Decryption

(ii) Cipher Block Chaining (CBC) Mode

CBC mode of operation provides message dependence for generating cipher text and makesthe system non-deterministic.

Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows:

- Load the n-bit Initialization Vector (IV) in the top register
- XOR the n-bit plaintext block with data value in top register
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed cipher text block into top register and continue the operation till all plaintext blocks are processed
- For decryption, IV data is XORed with first cipher text block decrypted. The first cipher text block is also fed into to register replacing IV for decrypting next cipher text block

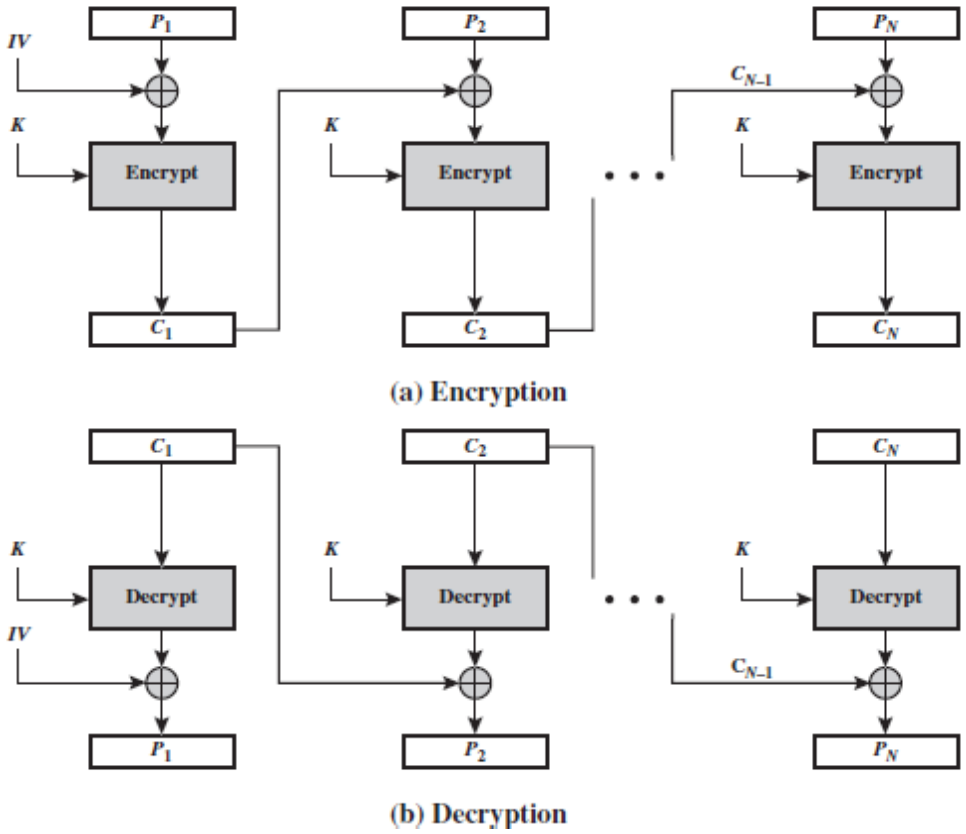
$$C_j = \mathrm{E}(K, [C_{j-1} \oplus P_j])$$

$$\mathrm{D}(K, C_j) = \mathrm{D}(K, E(K, [C_{j-1} \oplus P_j]))$$

$$\mathrm{D}(K, C_j) = C_{j-1} \oplus P_j$$

$$C_{j-1} \oplus \mathrm{D}(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

CBC	$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}])$ $C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}])\ \ j = 2, \ldots, N$	$P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$ $P_j = \mathrm{D}(K, C_j) \oplus C_{j-1}\ \ \ j = 2, \ldots, N$



CBC - Encryption and Decryption

(iii) Cipher Feedback (CFB) Mode

In this mode, each cipher text block gets 'fed back' into the encryption process in order toencrypt the next plaintext block.

Operation

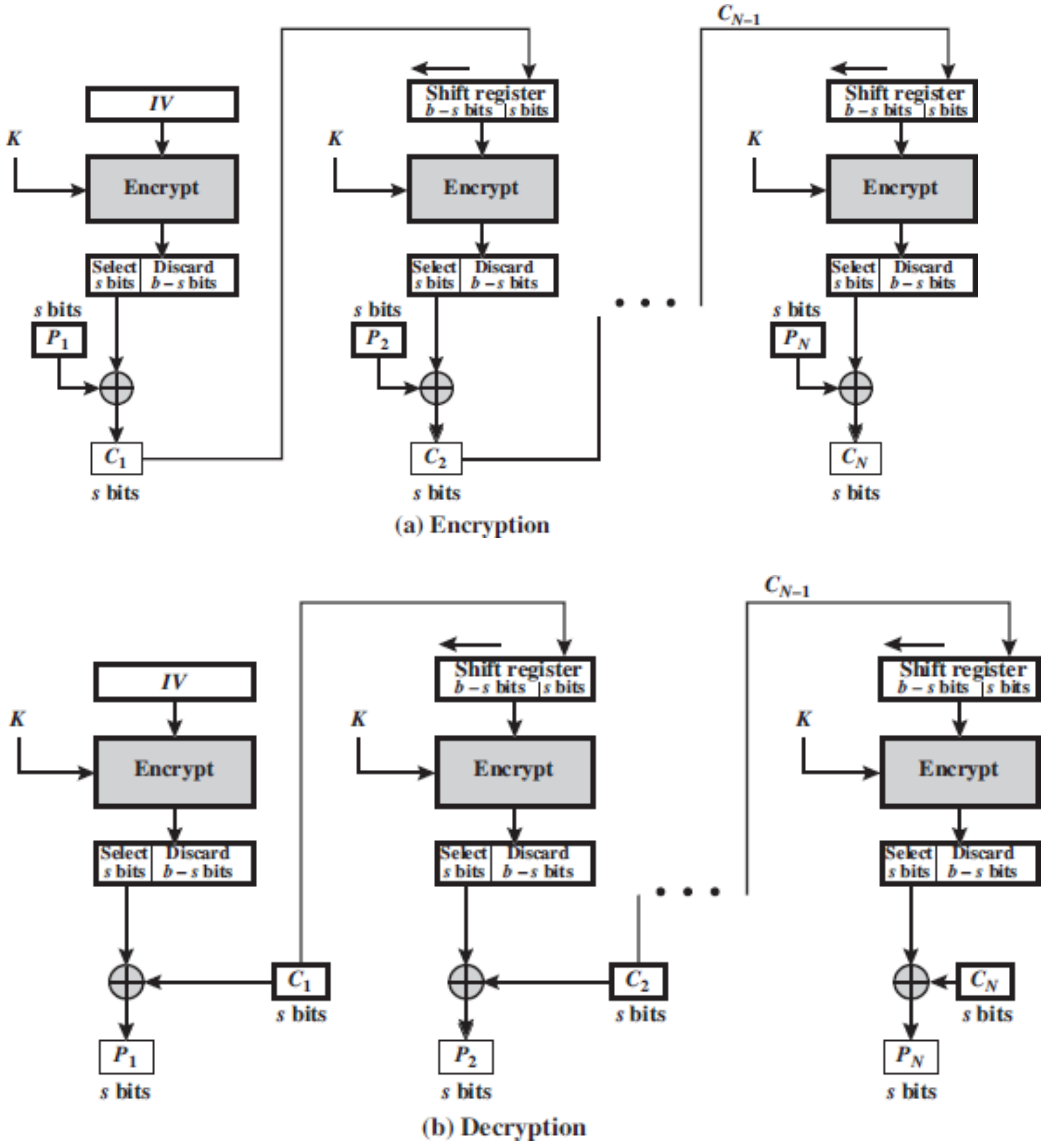
The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size 's' bits where 1 < s < n. The CFB mode requires an initialization vector (IV) as the initial random n-bit input block. The IV need not be secret. Steps of operation are:

- Load the IV in the top register
- Encrypt the data value in top register with underlying block cipher with key K
- Take only 's' number of most significant bits (left bits) of output of encryption process and XOR them with 's' bit plaintext message block to generate cipher text block
- Feed cipher text block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed
- Essentially, the previous cipher text block is encrypted with the key, and then the result is XORed to the current plaintext block
- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption

$$C_1 = P_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

$$P_1 = C_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

CFB	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \| C_{j-1}$ $j = 2, \ldots, N$ $O_j = \text{E}(K, I_j)$ $j = 1, \ldots, N$ $C_j = P_j \oplus \text{MSB}_s(O_j)$ $j = 1, \ldots, N$	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \| C_{j-1}$ $j = 2, \ldots, N$ $O_j = \text{E}(K, I_j)$ $j = 1, \ldots, N$ $P_j = C_j \oplus \text{MSB}_s(O_j)$ $j = 1, \ldots, N$



CFB - Encryption and Decryption

(iv) Output Feedback (OFB) Mode

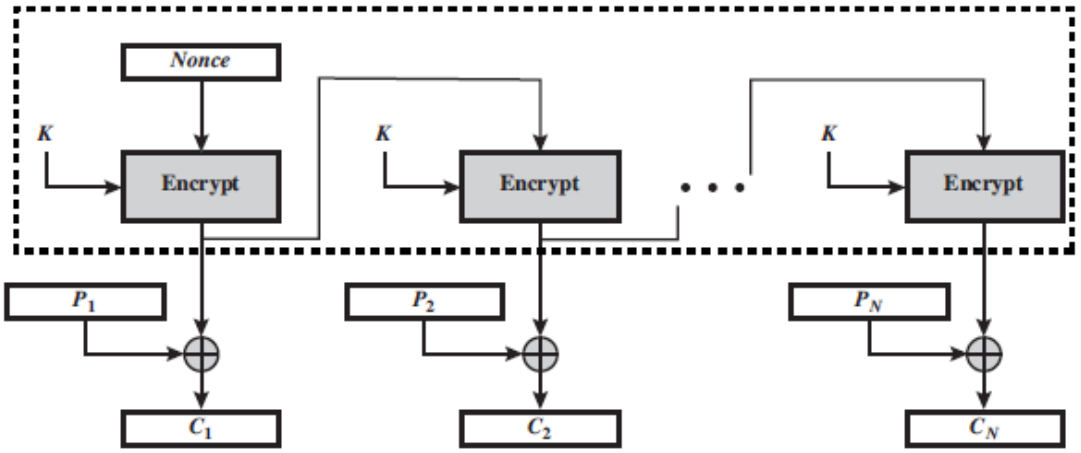
It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

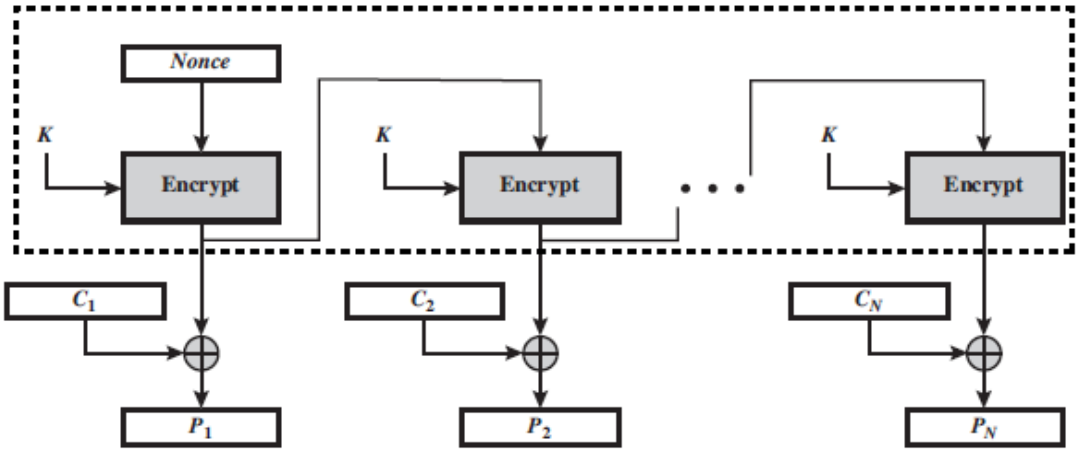
The operation is depicted in the following illustration:

$$C_j = P_j \oplus \text{E}(K, [C_{j-i} \oplus P_{j-1}])$$
$$P_j = C_j \oplus \text{E}(K, [C_{j-1} \oplus P_{j-1}])$$

OFB	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$	$I_1 = Nonce$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$



(a) Encryption



(b) Decryption

OFB - Encryption and Decryption

(v) Counter (CTR) Mode

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a cipher text block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

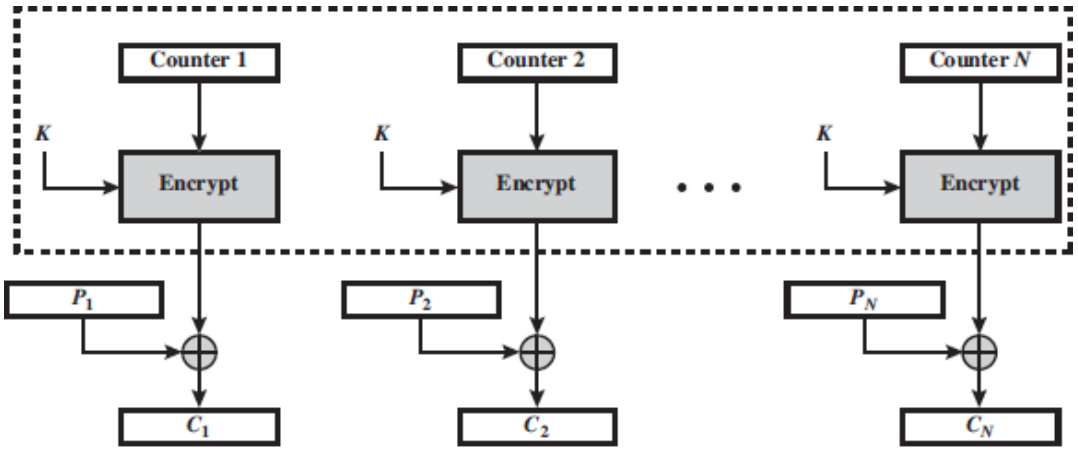
Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are:

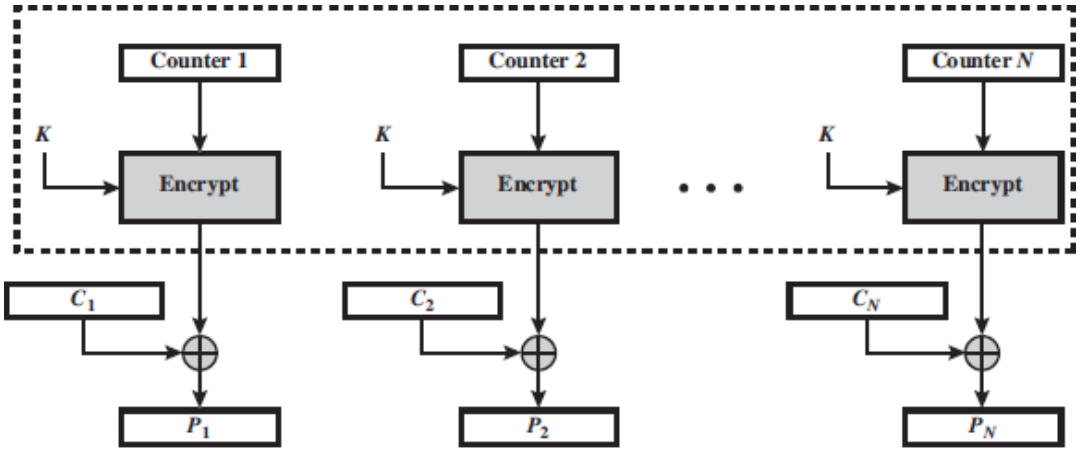
- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode
- Encrypt the contents of the counter with the key and place the result in the bottom register

- Take the first plaintext block P1 and XOR this to the contents of the bottom register
- The result of this is C1. Send C1 to the receiver and update the counter. The counter
- update replaces the cipher text feedback in CFB mode
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The cipher text block is XORed with the output of encrypted contents of counter value. After decryption of each cipher text block counter is updated as in case of encryption

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$



(a) Encryption



(b) Decryption

CTR - Encryption and Decryption

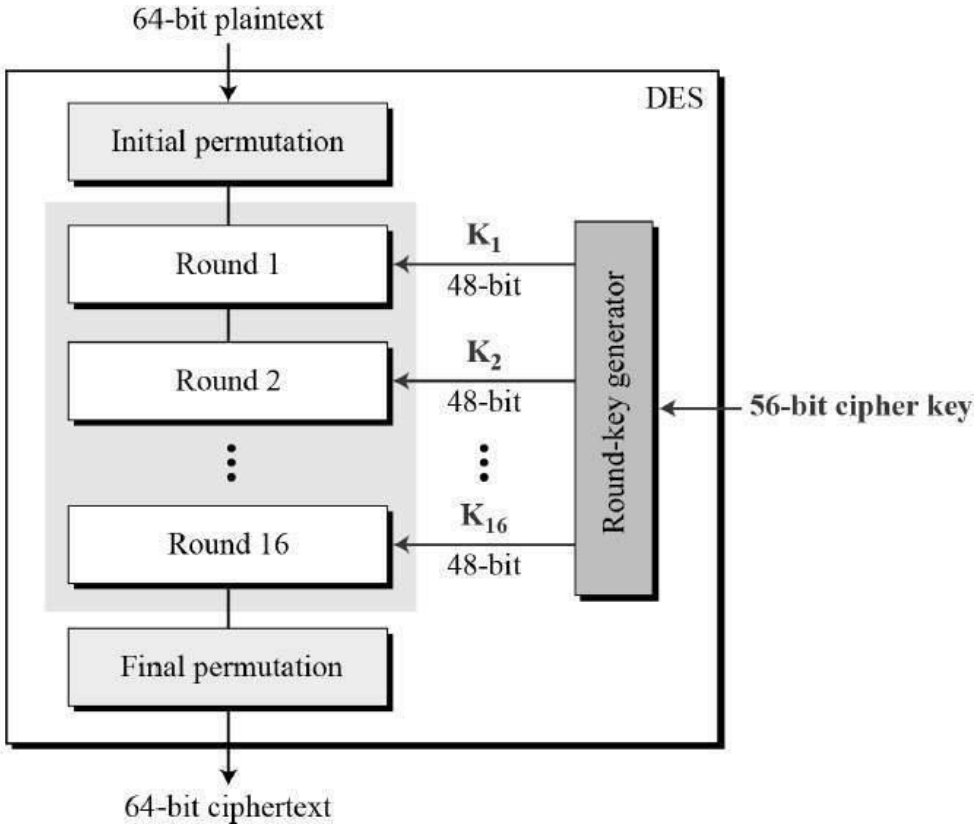
Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

DATA ENCRYPTION STANDARD (DES) ALGORITHM

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

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration:



DES Structure

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

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

Initial and Final Permutation

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

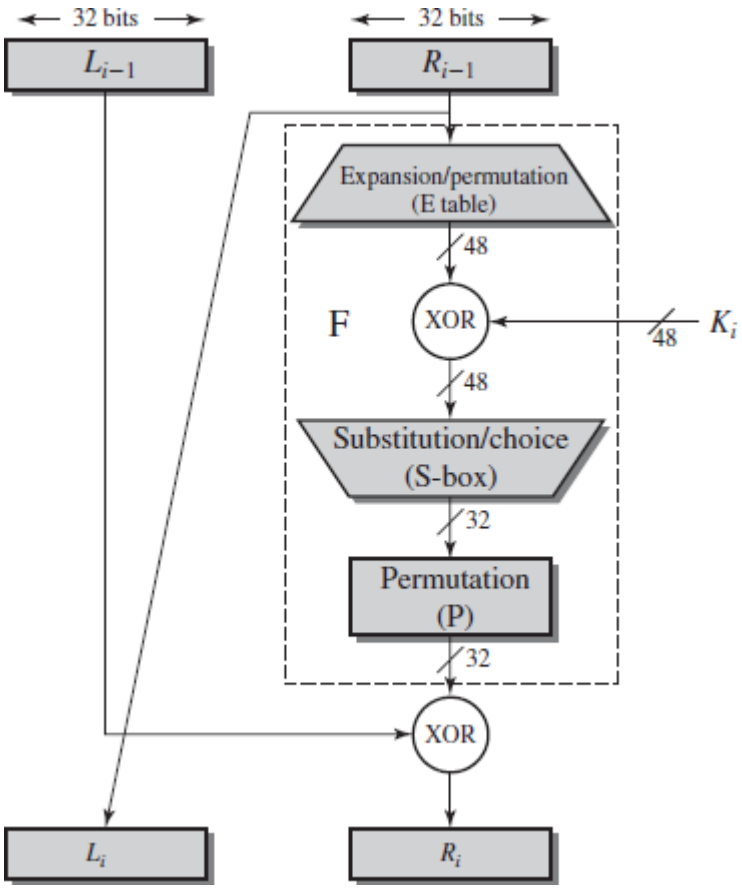
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Initial Permutation

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

Final Permutation

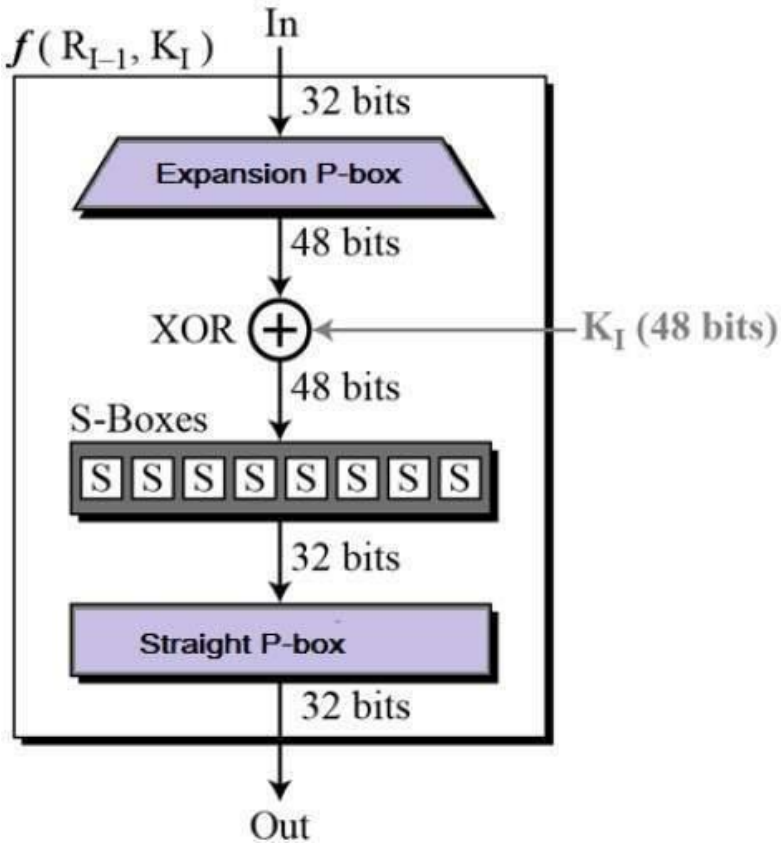
Details of one round in DES



One Round in DES

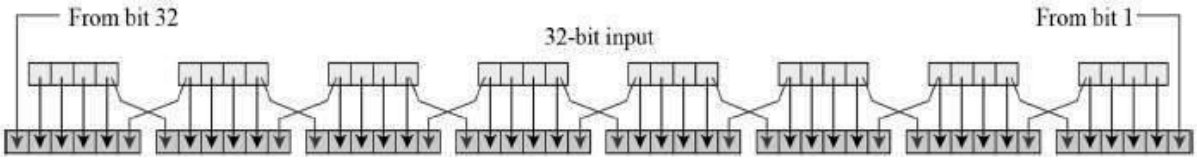
Round Function (F):

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



Round function

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



Expansion Permutation

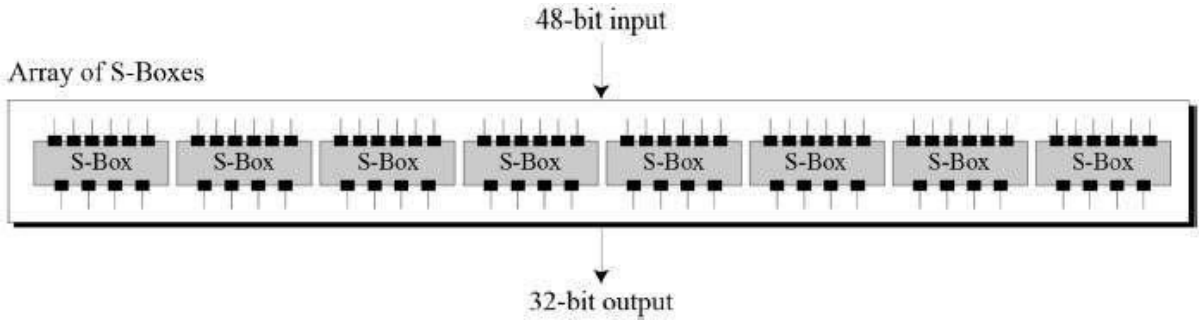
- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown:

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

Expansion Permutation Table

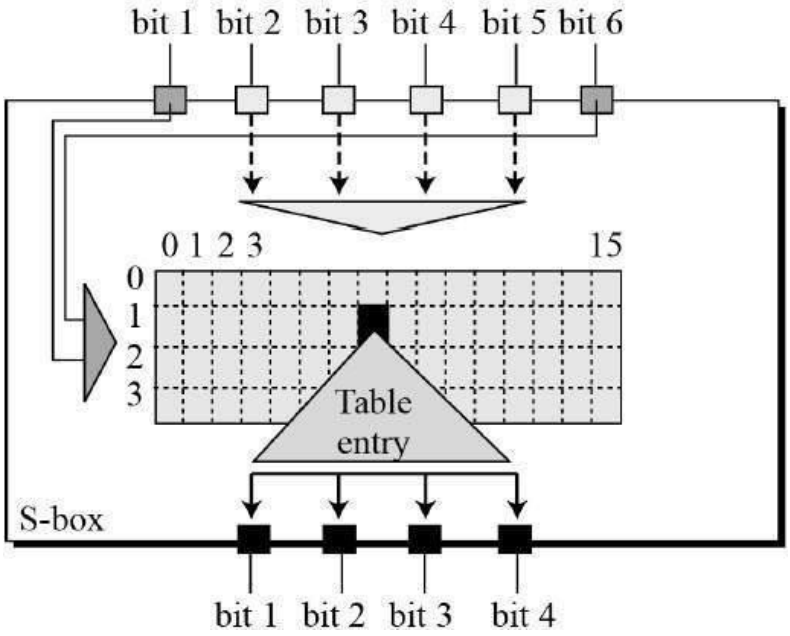
- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration



Substitution

The S-box rule is illustrated below



S-box rule

- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32-bit section.

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

S-boxes

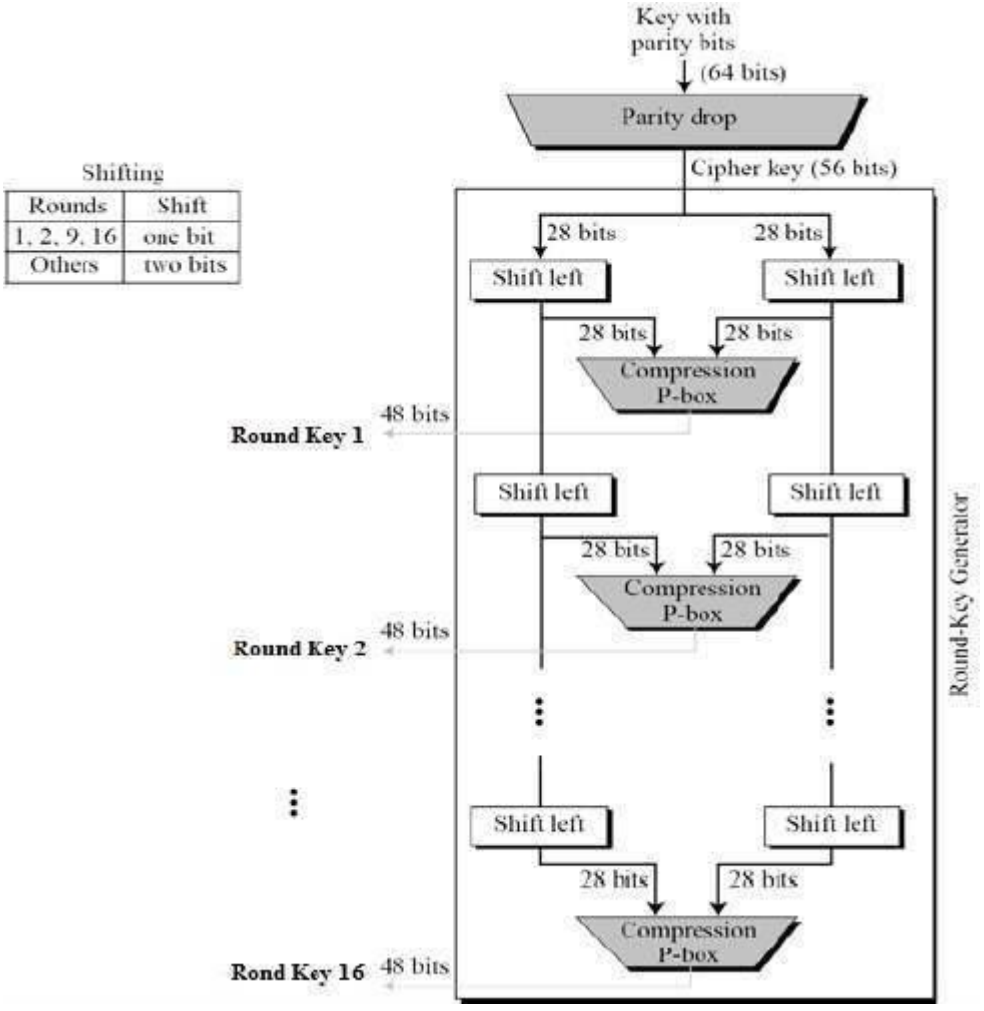
- **Straight Permutation** − The 32-bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Permutation Table

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration



Key Generation

ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

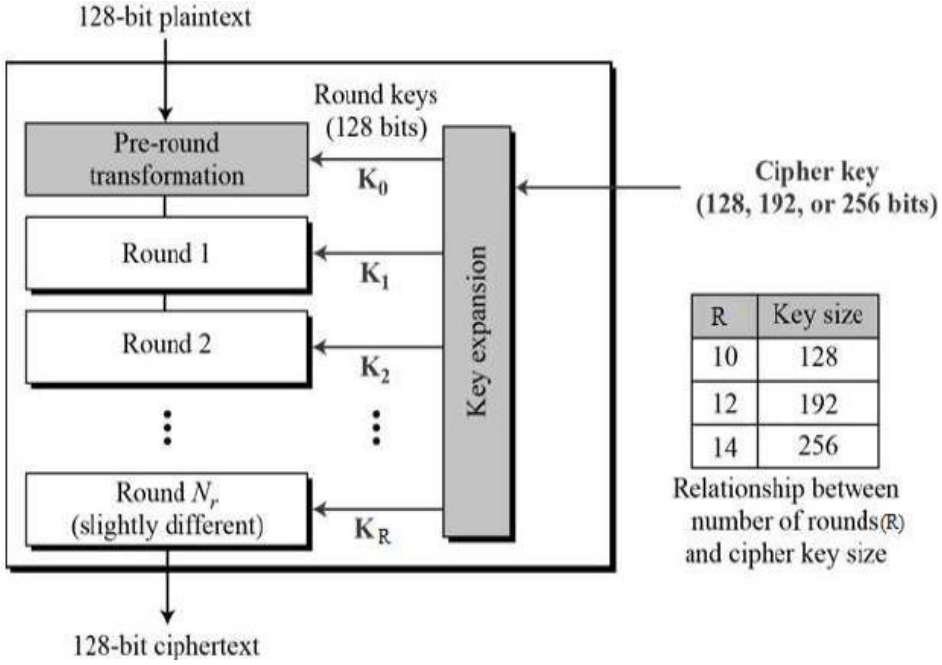
The features of AES are as follows:

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix: Unlike DES, the number of rounds in AES is variable and depends on the length of the key.

AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. The schematic of AES structure is given in the following illustration:



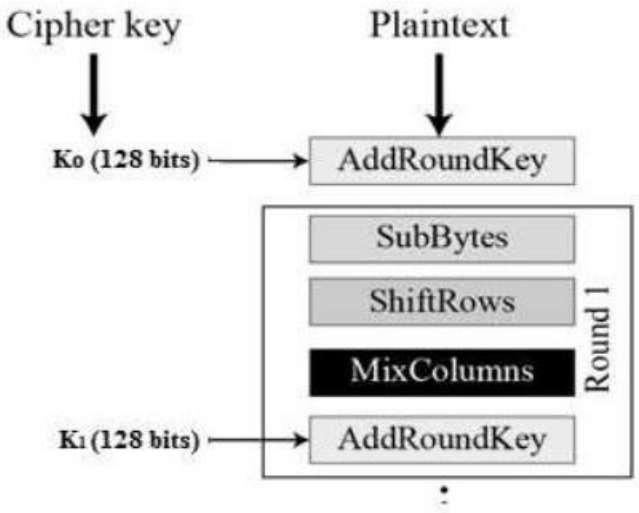
R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

AES Structure

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprises of four sub-processes. The first-round process is depicted below:



One round in AES

Byte Substitution (Sub Bytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

This stage (known as Sub Bytes) is simply a table lookup using a 16 × 16 matrix of byte values called an s-box.

 This matrix consists of all the possible combinations of (16× 16 = 256).

 However, the s-box is not just a random permutation of these values and there is a well defined method for creating the s-box tables.

Shift rows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are reinserted on the right side of row. Shift is carried out as follows:

- First row is not shifted
- Second row is shifted one (byte) position to the left
- Third row is shifted two positions to the left
- Fourth row is shifted three positions to the left
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other

Mix Columns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Add round key

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the cipher text. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES cipher text is similar to the encryption process in thereverse order. Each round consists of the four processes conducted in the reverse order:

- Add round key
- Mix columns
- Shift rows
- Byte substitution

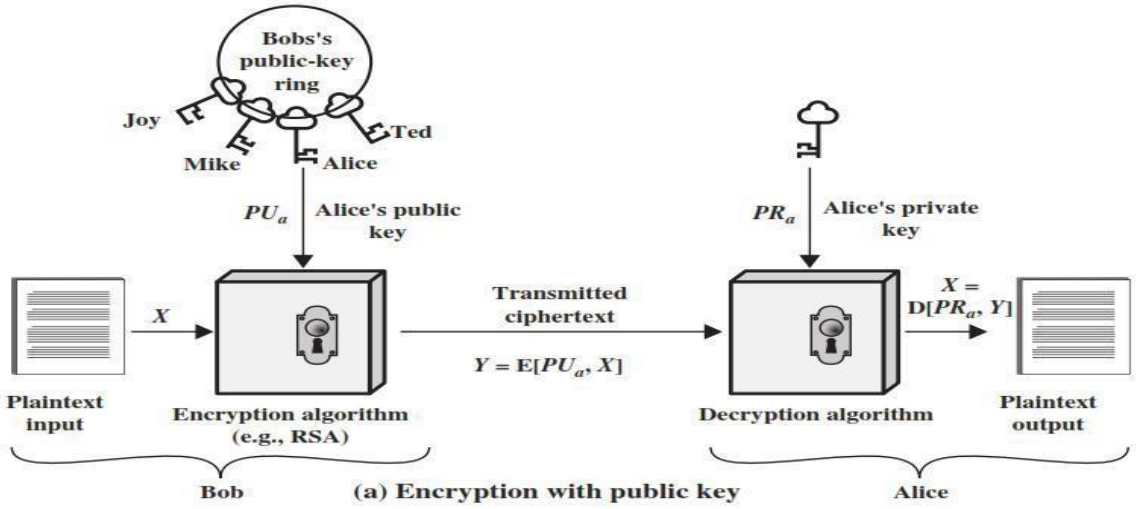
Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related

PUBLIC KEY CRYPTO SYSTEM (OR) ASYMMETRIC ENCRYPTION MODEL

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of key distribution, and the second one related to digital signature. Asymmetric algorithms is a type of cryptosystems in which encryption can be performed by one key (Public Key) and decryption can be performed by another key (Private Key). These algorithms have the following important characteristic.

It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key. A public-key encryption scheme has six ingredients

Plaintext: This is the readable message or data that is fed into the algorithm as input.



Encryption with Public key

Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.

• Public and private keys: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.

• Cipher text: 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 cipher texts.

• Decryption algorithm: This algorithm accepts the cipher text and the matching key and produces the original plaintext.

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

2. 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. suggests, each user maintains a collection of public keys obtained from others.

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

4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key. With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed.

As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

RSA ALGORITHM:

The Rivest-Shamir- Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

The RSA scheme is a block cipher in which the plaintext and cipher text are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 21024. We examine RSA in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and crypt analytical implications of RSA

Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to log2 (n) + 1; in practice, the block size is i bits, where $2i < n \leq 2i+1$. Encryption and decryption are of the following form, for some plaintext block M and cipher text block C.

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

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}.

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

1. It is possible to find values of *e*, *d*, *n* such that $Med \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $Me \bmod n$ and $Cd \bmod n$ for all values of $M < n$.
3. It is infeasible to determine *d* given *e* and *n*.

We need to find a relationship of the form
$M^{ed} \bmod n = M$

The preceding relationship holds if *e* and *d* are multiplicative inverses modulo $\varphi(n)$, where $\varphi(n)$ is the Euler totient function.

For p,q prime, $\varphi(pq) = (p - 1)(q - 1)$. The relationship between e and d can be expressed as
$$ed \bmod \varphi(n) = 1$$
This is equivalent to saying
$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$
That is, e and d are multiplicative inverses mod $\varphi(n)$. Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\varphi(n)$. Equivalently, $\gcd(\varphi(n), d) = 1$.

The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

The private key consists of $\{d, n\}$ and the public key consists of $\{e, n\}$. Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates $C = M^e \bmod n$ and transmits C. On receipt of this cipher text, user A decrypts by calculating $M = C^d \bmod n$.

RSA algorithm

Key Generation Alice

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

Encryption by Bob with Alice's Public Key

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

Decryption by Alice with Alice's Public Key

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

Example of RSA algorithm

An example of RSA can be given as,

Select primes: p=17 & q=11

Compute $n = pq$ =17×11=187

Compute ø(n)=(p–1)(q-1)=16×10=160

Select e: gcd(e,160)=1; choose e=7

Determine d*: de*=1 mod 160 and d < 160 Value is d=23 since 23×7=161= 10×160+1

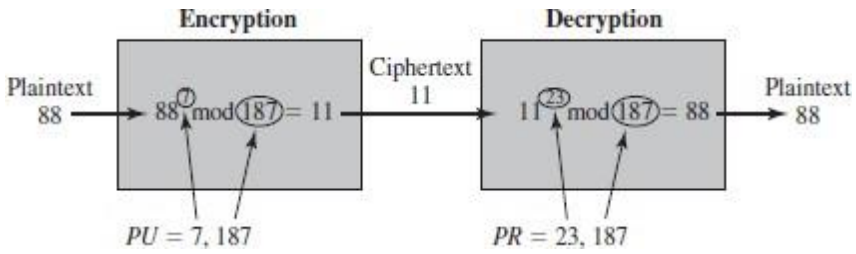
Publish public key KU={7,187}

Keep secret private key KR= {23,187}

Now, given message M = 88 (nb. 88<187)

Encryption: C = 887 mod 187 = 11

Decryption: M = 1123 mod 187 = 88



DIFFIE-HEILMAN KEY EXCHANGE/AGREEMENT ALGORITHM:

Whitefield Diffie and Martin Hellman devised an amazing solution to the problem of key agreement, or key exchange in 1976. This solution is called as the Diffie-Hellman Key Exchange /Agreement Algorithm. The beauty of this scheme is that the two parties, who want to communicate securely, can agree-on a symmetric key using this technique.

It might come as a surprise, but Kl is actually equal to K2! This means that Kl =K2 = K is the symmetric key, which Alice and Bob must keep secret and can henceforth usefor encrypting/decrypting their messages with. The mathematics behind this is quite interesting.

Let us try to understand what this actually means, in simple terms. (a) Firstly, take a look at what Alice does in step 6. Here, Alice computes: K1 = BX mod n. What is B? From step 4, we have: B = gYmod n. Therefore, if we substitute this value of B in step 6, we will have the following equation: Kl = (g)yr mod n =(g)YXmod n. (b) Now, take a look at

what Bob does in step 7. Here, Bob computes: K2 = AY mod n. What is A? From step 2, we have: A = gX mod n. Therefore, if we substitute this value of A in step 7, we will have the following equation: K2 = (gX)Y mod n = gXY' mod n. Now, basic mathematics says that: KYX = KXY Therefore, in this case, we have: K I = K2 = K.

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

Diffie- Hellman Key Exchange algorithm

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
   Let n = 11, g = 7.
2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$
   Let x = 3. Then, we have, $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$.
3. Alice sends the number A to Bob.
   Alice sends 2 to Bob.
4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$
   Let y = 6. Then, we have, $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$.
5. Bob sends the number B to Alice.
   Bob sends 4 to Alice.
6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$
   We have, $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$.
7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$
   We have, $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$.

Example of Diffie-Hellman Key exchange

UNIT-3

Message Authentication Algorithms and Hash Function: Authentication Requirements, Functions, Message Authentication Codes, Hash Functions, Secure Hash Algorithms, HMAC, CMAC, Digital Signatures, , ***Authentication Applications:*** *Kerberos, X.509 Authentication Services, Public-Key Infrastructure.*

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. It is intended against the attacks like content modification, sequence modification, timing modification and repudiation. For repudiation, concept of digital signatures is used to counter it. There are three classes by which different types of functions that may be used to produce an authenticator. They are:

- ***Message encryption***–the ciphertext serves as authenticator
- ***Message authentication code (MAC)***–a public function of the message and a secret key producing a fixed-length value to serve as authenticator. This does not provide a digital signature because A and B share the same key.
- ***Hash function***–a public function mapping an arbitrary length message into a fixed- length hash value to serve as authenticator. This does not provide a digital signature because there is no key.

MESSAGE ENCRYPTION:

Message encryption by itself can provide a measure of authentication. The analysis differs for conventional and public-key encryption schemes. The message must have come from the sender itself, because the ciphertext can be decrypted using his (secret or public) key. Also, none of the bits in the message have been altered because an opponent does not know how to manipulate the bits of the ciphertext to induce meaningful changes to the plaintext. Often one needs alternative authentication schemes than just encrypting the message.

- Sometimes one needs to avoid encryption of full messages due to legal requirements.

Encryption and authentication may be separated in the system architecture.

The different ways in which message encryption can provide authentication, confidentiality in both symmetric and asymmetric encryption techniques is explained with the table below:

Confidentiality and Authentication Implications of Message Encryption

A → B: $E_K[M]$ •Provides confidentiality —Only A and B share K •Provides a degree of authentication —Could come only from A —Has not been altered in transit —Requires some formatting/redundancy •Does not provide signature —Receiver could forge message —Sender could deny message (a) Symmetric encryption
A → B: $E_{KU_b}[M]$ •Provides confidentiality —Only B has KR_b to decrypt •Provides no authentication —Any party could use KU_b to encrypt message and claim to be A (b) Public-key encryption: confidentiality
A → B: $E_{KR_a}[M]$ •Provides authentication and signature —Only A has KR_a to encrypt —Has not been altered in transit —Requires some formatting/redundancy —Any party can use KU_a to verify signature (c) Public-key encryption: authentication and signature
A → B: $E_{KU_b}[E_{KR_a}(M)]$ •Provides confidentiality because of KU_b •Provides authentication and signature because of Kr_a (d) Public-key encryption: confidentiality, authentication, and signature

MESSAGE AUTHENTICATION CODE

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as cryptographic checksum or MAC, which is appended to the message. This technique assumes that both the communicating parties say A and B share a common secret key K. When A has a message to send to B, it calculates MAC as a function C of key and message given as: **MAC=Ck(M)** The message

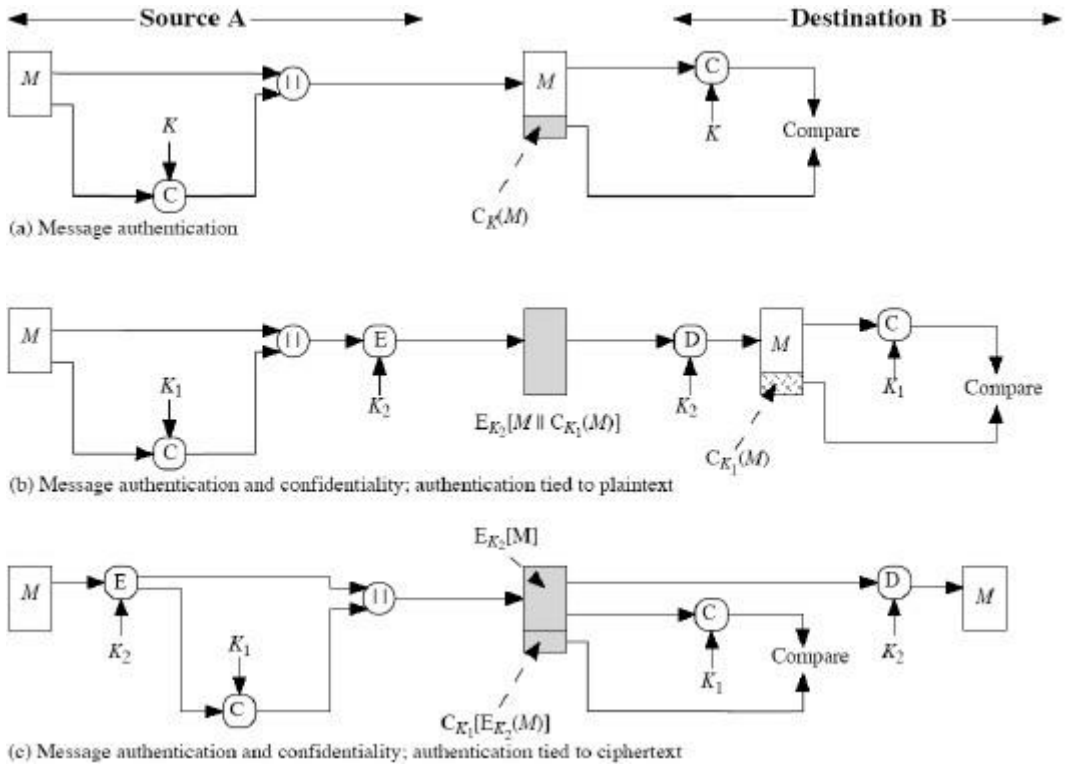
and the MAC are transmitted to the intended recipient, who upon receiving performs the same calculation on the received message, using the same secret key to generate a new MAC. The received MAC is compared to the calculated MAC and only if they match, then:

1. The receiver is assured that the message has not been altered: Any alternations beendone the MAC's do not match.

2. The receiver is assured that the message is from the alleged sender: No one except the sender has the secret key and could prepare a message with a proper MAC.

3. If the message includes a sequence number, then receiver is assured of propersequence as an attacker cannot successfully alter the sequence number.

Basic uses of Message Authentication Code (MAC) are shown in the figure:



There are three different situations where use of a MAC is desirable:

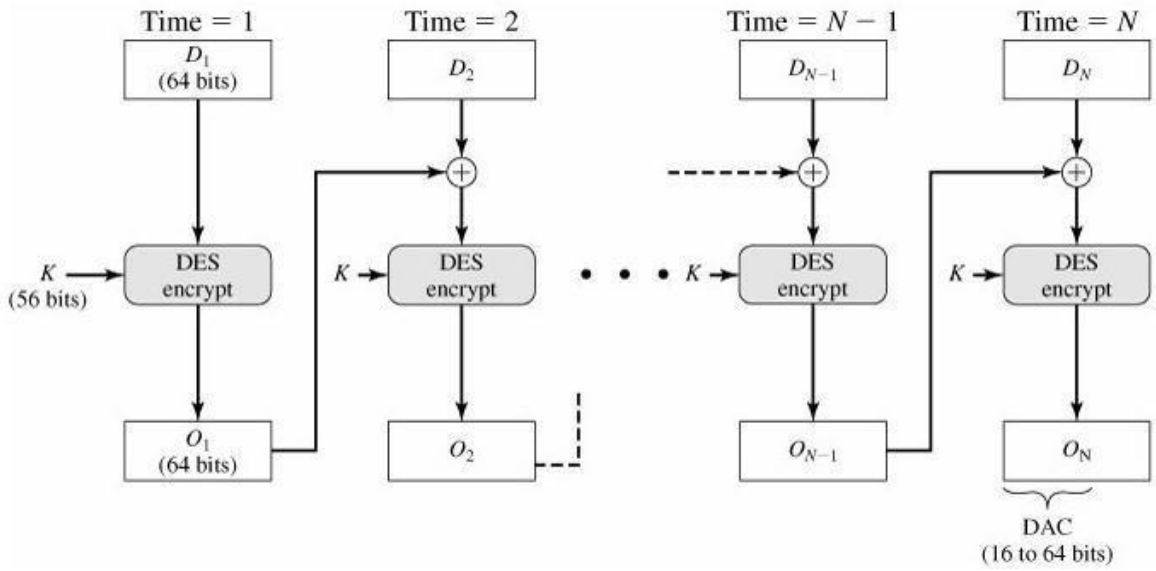
If a message is broadcast to several destinations in a network (such as a military control center), then it is cheaper and more reliable to have just one node responsible to evaluate the authenticity –message will be sent in plain with an attached authenticator.

If one side has a heavy load, it cannot afford to decrypt all messages –it will just checkthe authenticity of some randomly selected messages.

Authentication of computer programs in plaintext is very attractive service as they need not be decrypted every time wasting of processor resources. Integrity of the program can always be checked by MAC.

MESSAGE AUTHENTICATION CODE BASED ON DES

The Data Authentication Algorithm, based on DES, has been one of the most widely used MACs for a number of years. The algorithm is both a FIPS publication (FIPS PUB 113) andan ANSI standard (X9.17). But, security weaknesses in this algorithm have been discovered and it is being replaced by newer and stronger algorithms. The algorithm canbe defined as using the cipher block chaining (CBC) mode of operation of DES shown below with an initialization vector of zero.



The data (e.g., message, record, file, or program) to be authenticated are grouped into contiguous 64-bit blocks: D1, D2,..., DN. If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm, E, and a secret key, K, a data authentication code (DAC) is calculated as follows:

$O_1 = E(K, D_1)$

$O_2 = E(K, [D_2 \oplus O_1])$

$O_3 = (K, [D_3 \oplus O_2])$

•

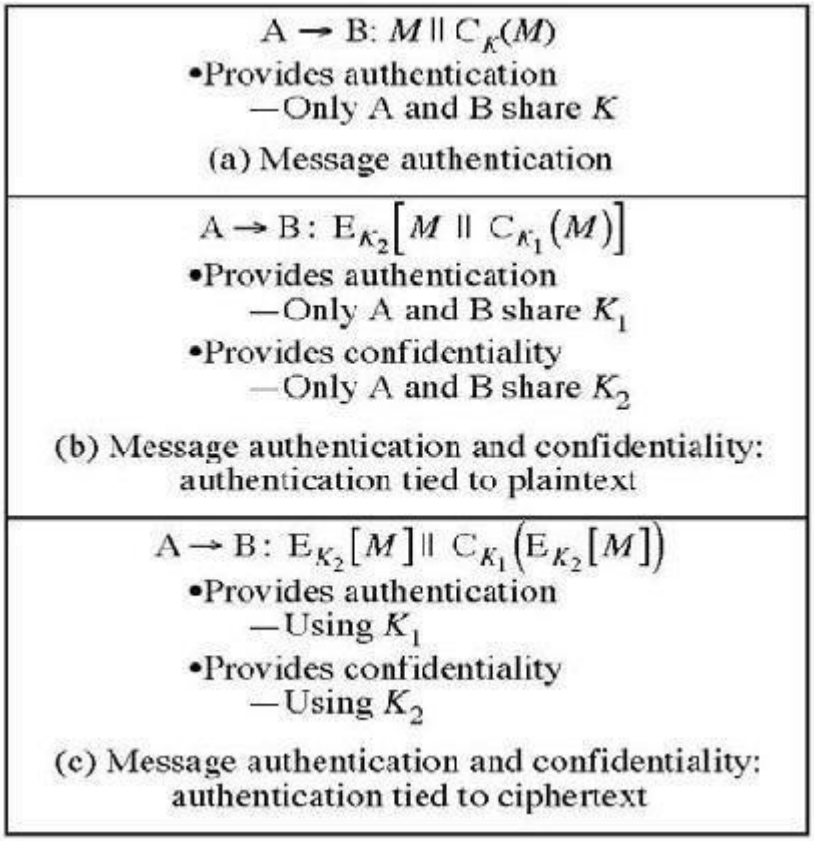
•

•

$O_N = E(K, [D_N \oplus O_{N1}])$

The DAC consists of either the entire block ON or the leftmost M bits of the block, with 16 ≤ M ≤ 64

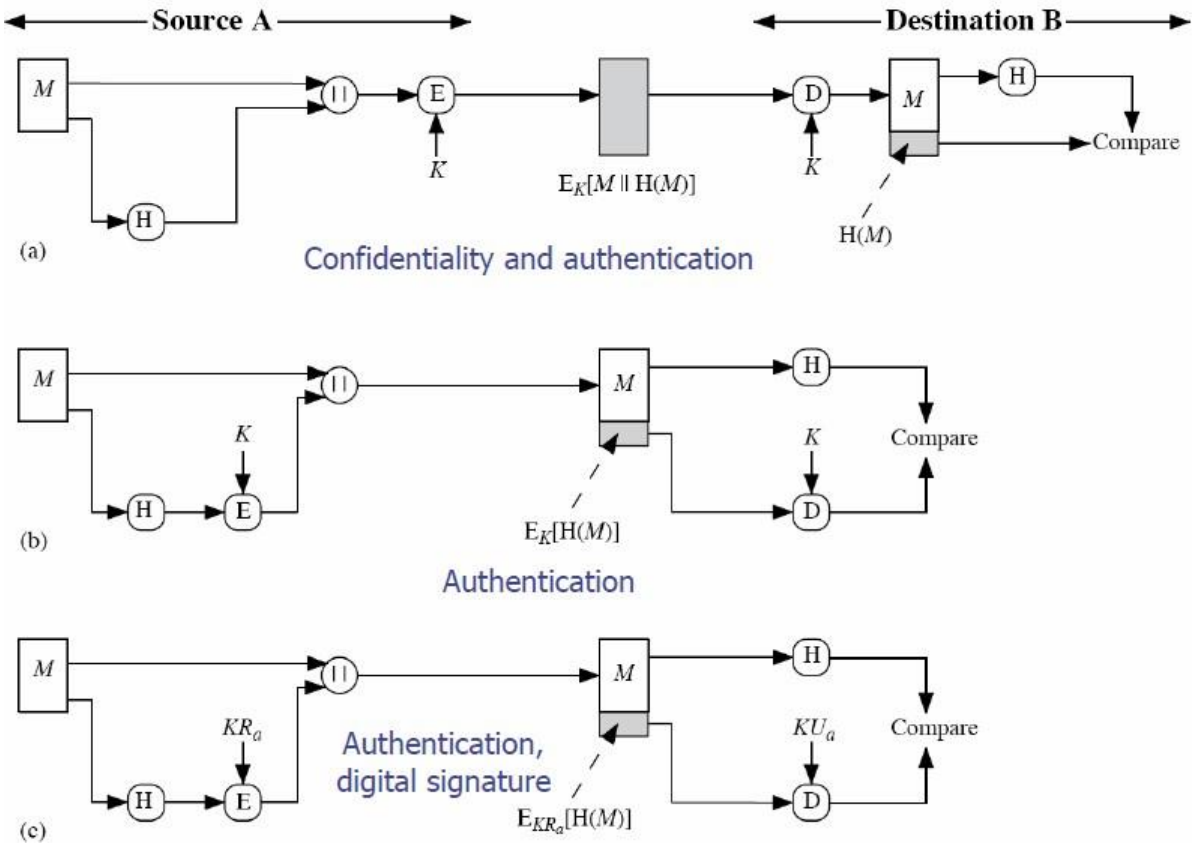
Use of MAC needs a shared secret key between the communicating parties and also MAC does not provide digital signature. The following table summarizes the confidentiality and authentication implications of the approaches shown above.

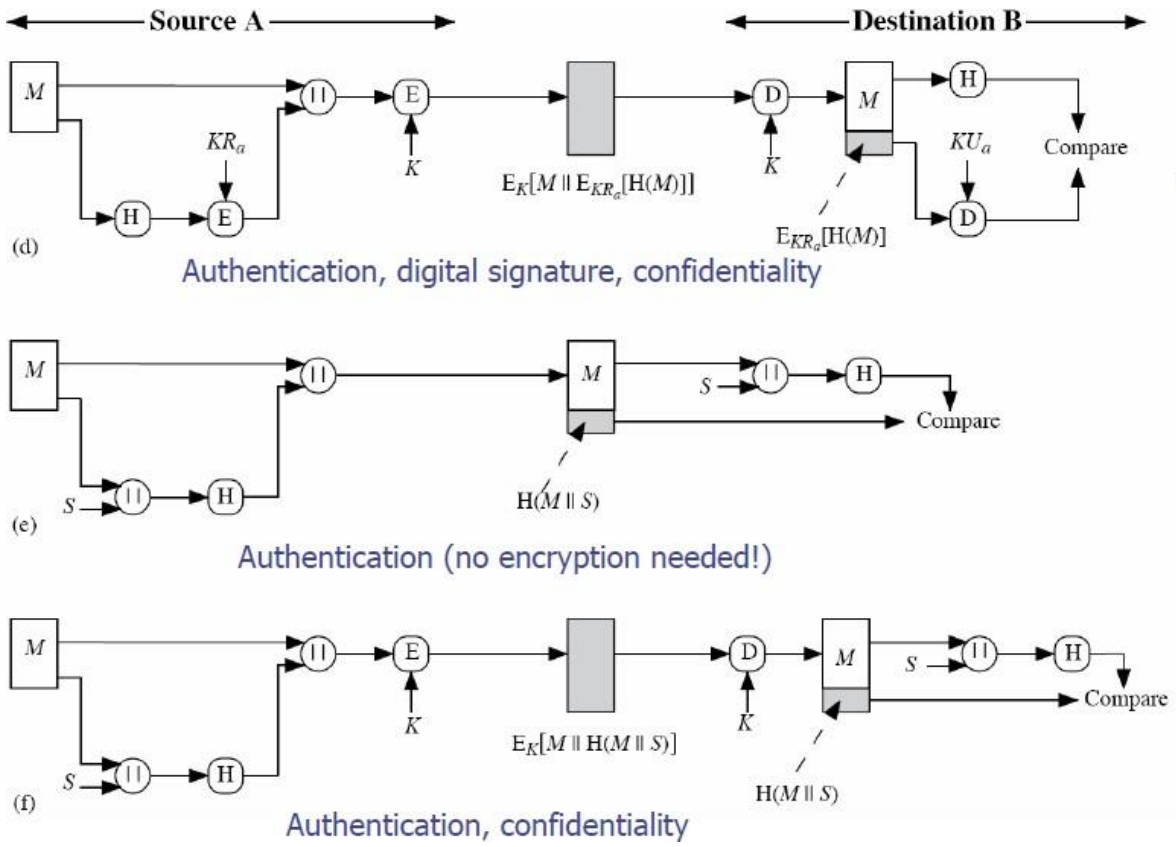


HASH FUNCTION

A variation on the message authentication code is the one-way hash function. As with the message authentication code, the hash function accepts a variable-size message M as input and produces a fixed-size hash code H(M), sometimes called a message digest, as output. The hash code is a function of all bits of the message and provides an error- detection capability: A change to any bit or bits in the message results in a change to the hash code. A variety of ways in which a hash code can be used to provide message authentication is shown below and explained stepwise in the table.

$A \rightarrow B: E_K[M \parallel H(M)]$ •Provides confidentiality —Only A and B share K •Provides authentication —H(M) is cryptographically protected (a) Encrypt message plus hash code	$A \rightarrow B: E_K[M \parallel E_{KR_a}[H(M)]]$ •Provides authentication and digital signature •Provides confidentiality —Only A and B share K (d) Encrypt result of (c) - shared secret key
$A \rightarrow B: M \parallel E_K[H(M)]$ •Provides authentication —H(M) is cryptographically protected (b) Encrypt hash code - shared secret key	$A \rightarrow B: M \parallel H(M \parallel S)$ •Provides authentication —Only A and B share S (e) Compute hash code of message plus secret value
$A \rightarrow B: M \parallel E_{KR_a}[H(M)]$ •Provides authentication and digital signature —H(M) is cryptographically protected —Only A could create $E_{KR_a}[H(M)]$ (c) Encrypt hash code - sender's private key	$A \rightarrow B: E_K[M \parallel H(M) \parallel S]$ •Provides authentication —Only A and B share S •Provides confidentiality —Only A and B share K (f) Encrypt result of (c)





In cases where confidentiality is not required, methods b and c have an advantage over those that encrypt the entire message in that less computation is required. Growing interest for techniques that avoid encryption is due to reasons like, Encryption software is quite slow and may be covered by patents. Also encryption hardware costs are not negligible and the algorithms are subject to U.S export control. A fixed-length hash value h is generated by a function H that takes as input a message of arbitrary length**: h=H(M).**

A sends M and H(M)

B authenticates the message by computing H(M) and checking the match

Requirements for a hash function: The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be used for message authentication, the hash function H must have the following properties

H can be applied to a message of any size

H produces fixed-length output

Computationally easy to compute H(M) for any given M

Computationally infeasible to find M such that H(M)=h, for a given h, referred to as the *one-way property*

Computationally infeasible to find M' such that H(M')=H(M), for a given M, referred to as *weak collision resistance*.

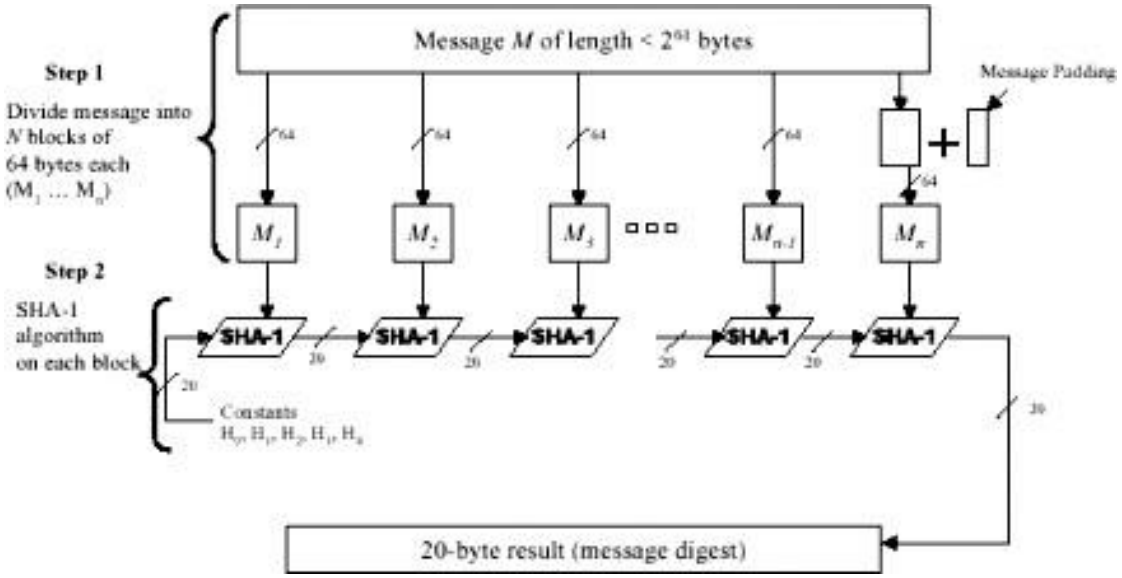
Computationally infeasible to find M,M' with H(M)=H(M') (to resist to birthday attacks), referred to as *strong collision resistance*.

Examples of simple hash functions are:

- Bit-by-bit XOR of plaintext blocks: h= D1⊕D2⊕…⊕DN
- Rotated XOR –before each addition the hash value is rotated to the left with 1 bit
- Cipher block chaining technique without a secret key.

SECURE HASH ALGORITHM

The secure hash algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST). SHA-1 is the best established of the existing SHA hash functions, and is employed in several widely used security applications and protocols. The algorithm takes as input a message with a maximum length of less than 264 bits and produces as output a 160-bit message digest.



The input is processed in 512-bit blocks. The overall processing of a message follows the structure of MD5 with block length of 512 bits and a hash length and chaining variable length of 160 bits. The processing consists of following steps:

1.) ***Append Padding Bits:*** The message is padded so that length is congruent to 448 modulo 512; padding always added –one bit 1 followed by the necessary number of 0 bits.

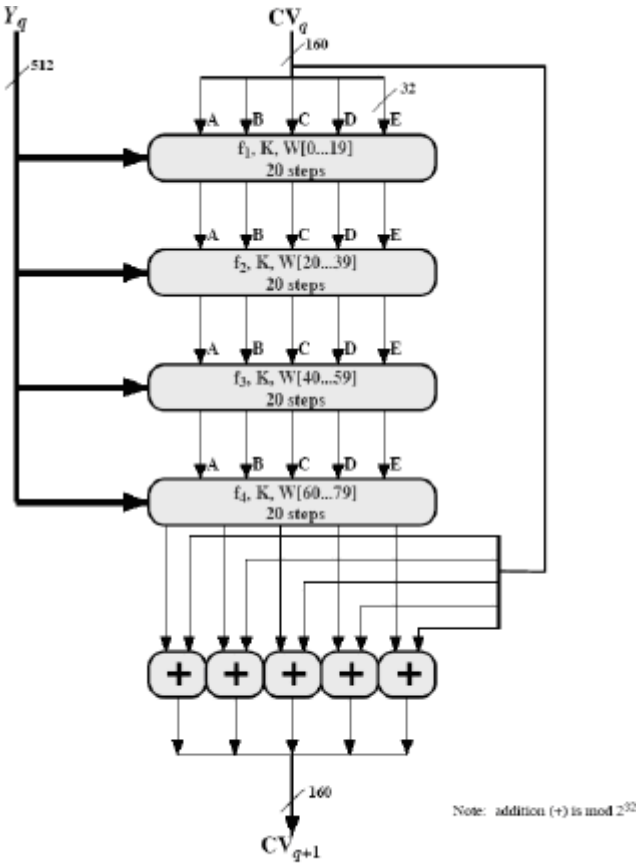
2.) ***Append Length:*** a block of 64 bits containing the length of the original message is added.

3.) ***Initialize MD buffer*****:** A 160-bit buffer is used to hold intermediate and final results on the hash function. This is formed by 32-bit registers A,B,C,D,E. Initial values: A=0x67452301, B=0xEFCDAB89, C=0x98BADCFE, D=0x10325476, E=C3D2E1F0. Stores in big-endian format

i.e. the most significant bit in low address.

4.) ***Process message in blocks 512-bit (16-word) blocks*****:** The processing of a single 512-bit block is shown above. It consists of four rounds of processing of 20 steps each. These four rounds have similar structure, but uses a different primitive logical function, which we refer to as f1, f2, f3 and f4. Each round takes as input the current 512-bit block being processed and the 160-bit buffer value ABCDE and updates the contents of the buffer. Each round also makes use of four distinct additive constants Kt. The output of the fourth round i.e. eightieth step is added to the input to the first round to produce CVq+1.

5.) *Output:* After all L 512-bit blocks have been processed, the output from the Lth stage is the 160-bit message digest.

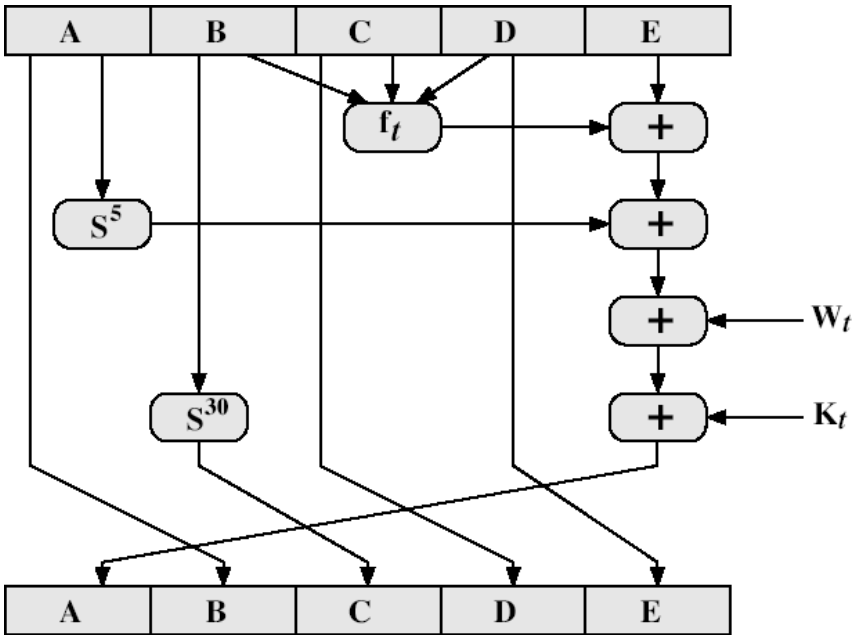


SHA-1 Processing of a Single 512-bit Block
(SHA-1 Compression Function)

The behavior of SHA-1 is as follows: **CV0 = IV CVq+1 = SUM32(CVq, ABCDEq) MD = CVL** Where, IV = initial value of ABCDE buffer ABCDEq = output of last round of processing of qth message block L = number of blocks in the message SUM32 = Addition modulo 232 MD = final message digest value.

SHA-1 Compression Function:

Each round has 20 steps which replaces the 5 buffer words. The logic present in each one of the 80 rounds present is given as **(A,B,C,D,E) <- (E + f(t,B,C,D) + S5(A)+ Wt+ Kt),A,S30(B),C,D** Where, A, B, C, D, E = the five words of the buffer t = step number; 0< t < 79 f(t,B,C,D) = primitive logical function for step t Sk = circular left shift of the 32-bit argument by k bits Wt = a 32-bit word derived from current 512-bit input block. Kt = an additive constant; four distinct values are used + = modulo addition.



SHA shares much in common with MD4/5, but with 20 instead of 16 steps in each of the 4 rounds. Note the 4 constants are based on sqrt(2,3,5,10). Note also that instead of just splitting the input block into 32-bit words and using them directly, SHA-1 shuffles and mixes them using rotates & XOR's to form a more complex input, and greatly increases the difficulty of finding collisions. A sequence of logical functions f0, f1,..., f79 is used in the SHA-1. Each f**t**, 0<=t<=79, operates on three 32-bit words B, C, D and produces a 32-bit word as output. ft(B,C,D) is defined as follows: for words B, C, D, **ft(B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19) ft(B,C,D) = B XOR C XOR D (20 <= t <= 39) ft(B,C,D) =(B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59) ft(B,C,D) = B XOR C XOR D (60**

<= t <= 79).

HMAC

Interest in developing a MAC, derived from a cryptographic hash code has been increasing mainly because hash functions are generally faster and are also not limited by export restrictions unlike block ciphers. Additional reason also would be that the library code for cryptographic hash functions is widely available. The original proposal is for incorporation of a secret key into an existing hash algorithm and the approach that received most support is HMAC. HMAC is specified as Internet standard RFC2104. It

makes use of the hash function on the given message. Any of MD5, SHA-1, RIPEMD-160 can be used.

HMAC Design Objectives

- To use, without modifications, available hash functions
- To allow for easy replaceability of the embedded hash functionTo
- preserve the original performance of the hash function
- To use and handle keys in a simple way
- To have a well understood cryptographic analysis of the strength of the MAC based onreasonable assumptions on the embedded hash function

The first two objectives are very important for the acceptability of HMAC. HMAC treats the hash function as a "black box", which has two benefits. First is that an existing implementation of the hash function can be used for implementing HMAC making the bulk of HMAC code readily available without modification. Second is that if ever an existing hash function is to be replaced, the existing hash function module is removed and new module is dropped in. The last design objective provides the main advantage of HMAC over other proposed hash-based schemes. HMAC can be proven secure provided that the embedded hash function has some reasonable cryptographic strengths.

Steps involved in HMAC algorithm:

1. Append zeroes to the left end of K to create a b-bit string K+ (ex: If K is of length 160-bits and b = 512, then K will be appended with 44 zero bytes).
2. XOR(bitwise exclusive-OR) K+ with ipad to produce the b-bit block Si.
3. Append M to Si.
4. Now apply H to the stream generated in step-3
5. XOR K+ with opad to produce the b-bit block S0.
6. Append the hash result from step-4 to S0.
7. Apply H to the stream generated in step-6 and output the result.

HMAC Algorithm

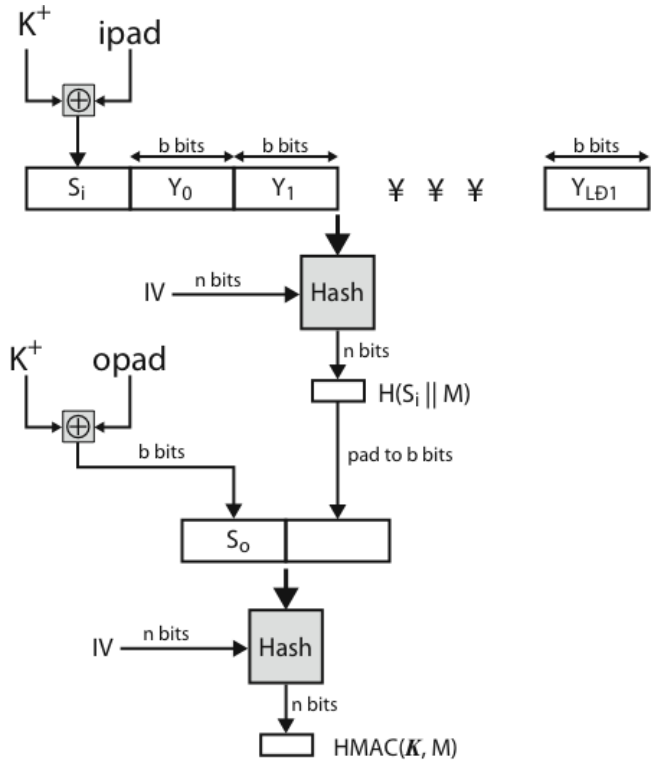
- **Define the following terms**

H = embedded hash function
M = message input to HMAC
Y_i = i^{th} block of M, $0 \le i \le L - 1$
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; if key length is greater than b, the key is input to the hash function to produce an n-bit key; recommended length ≥ n
K^+ = K padded with 0's on the left so that the result is b bits in length
ipad = 00110110 repeated b/8 times
opad = 01011100 repeated b/8 times

- **Then HMAC can be expressed as**

$$HMAC_K = H[\ (K^+ \oplus opad)\ ||\ H[K^+ \oplus ipad)\ ||\ M]\ ]$$

HMAC Structure



The XOR with ipad results in flipping one-half of the bits of K. Similarly, XOR with opad results in flipping one-half of the bits of K, but different set of bits. By passing Si and S0 through the compression function of the hash algorithm, we have pseudorandomly generated two keys from K.

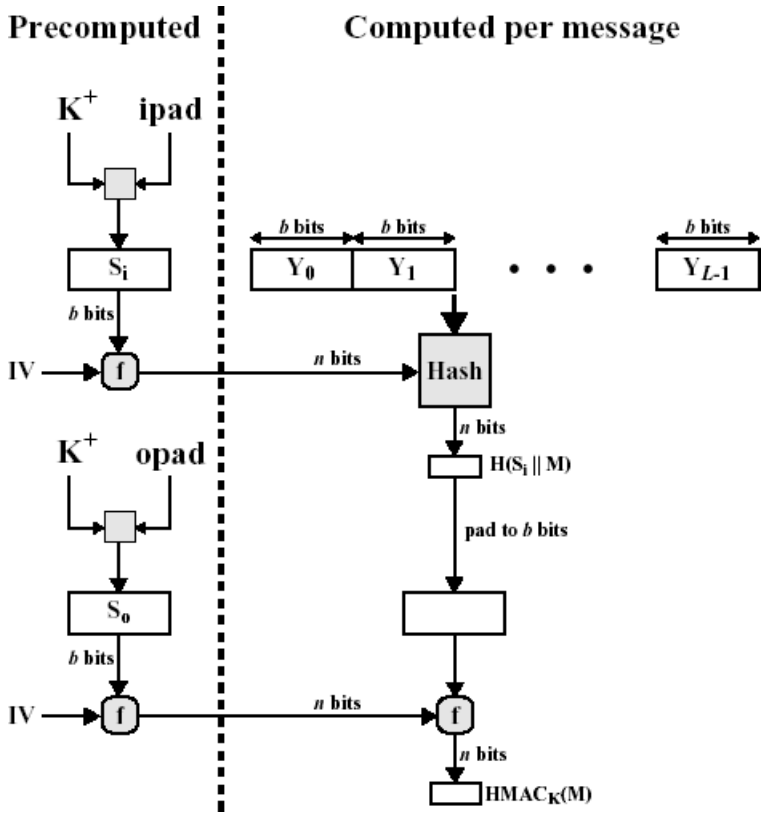
HMAC should execute in approximately the same time as the embedded hash function forlong messages. HMAC adds three executions of the hash compression function (for S0, Si, and the block produced from the inner hash)

A more efficient implementation is possible. Two quantities are precomputed.

f(IV, (K+ ⊕ ipad)

f(IV, (K+ ⊕ opad)

where f 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.



As shown in the above figure, the values are needed to be computed initially and every time a key changes. The precomputed quantities substitute for the initial value (IV) in the hash function. With this implementation, only one additional instance of the compressionfunction is added to the processing normally produced by the hash function. This implementation is worthwhile if most of the messages for which a MAC is computed are short.

Security of HMAC:

The appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC. 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-MAC pairs created with the same key. Have two classes of attacks on the embedded hash function:

1. 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.

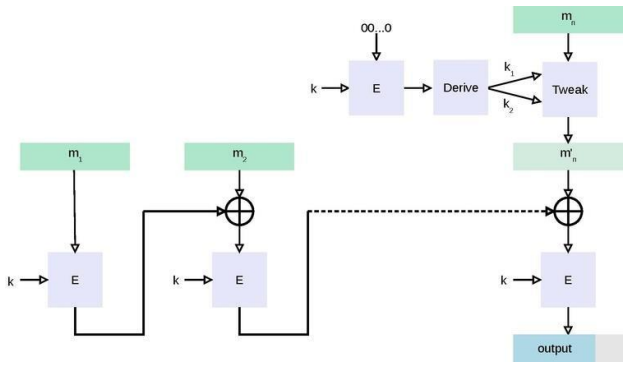
2. The attacker finds collisions in the hash function even when the IV is random and secret.

These attacks are likely to be caused by brute force attack on key used which has work of order 2n; or a birthday attack which requires work of order 2(n/2) - but which requires the attacker to observe 2n blocks of messages using the same key - very unlikely. So even MD5 is still secure for use in HMAC given these constraints.

CMAC

In cryptography, **CMAC** (Cipher-based Message Authentication Code) is a block cipher-based message authentication code algorithm. It may be used to provide assurance of the authenticity and, hence, the integrity of binary data. This mode of operation fixes security deficiencies of CBC-MAC (CBC-MAC is secure only for fixed-length messages).

The core of the CMAC algorithm is a variation of CBC-MAC that Black and Rogaway proposed and analyzed under the name XCBC and submitted to NIST.The XCBC algorithm efficiently addresses the security deficiencies of CBC-MAC, but requires three keys. Iwata and Kurosawa proposed an improvement of XCBC and named the resulting algorithm One-Key CBC-MAC (OMAC). They later submitted OMAC1, a refinement of OMAC, and additional security analysis. The OMAC algorithm reduces the amount of key material required for XCBC. CMAC is equivalent to OMAC1.



To generate an ℓ-bit CMAC tag (t) of a message (m) using a b-bit block cipher (E) and a secret key (k), one first generates two b-bit sub-keys ($k1$ and $k2$) using the following algorithm (this is equivalent to multiplication by x and x^2 in a finite field GF(2^b)). Let $\ll$ denote the standard left-shift operator and $\oplus$ denote exclusive or:

1. Calculate a temporary value $k0 = Ek(0)$.
2. If msb($k0$) = 0, then $k1 = k0 \ll 1$, else $k1 = (k0 \ll 1) \oplus C$; where C is a certain constant that depends only on b. (Specifically, C is the non-leading coefficients of the lexicographically first irreducible degree-b binary polynomial with the minimal number of ones.)
3. If msb($k1$) = 0, then $k2 = k1 \ll 1$, else $k2 = (k1 \ll 1) \oplus C$.
4. Return keys ($k1$, $k2$) for the MAC generation process.

As a small example, suppose $b = 4$, $C = 00112$, and $k0 = Ek(0) = 01012$. Then $k1 = 10102$ and $k2 = 0100 \oplus 0011 = 01112$.

The CMAC tag generation process is as follows:

1. Divide message into b-bit blocks $m = m1 \parallel ... \parallel mn{-}1 \parallel mn$ where $m1$, ..., $mn{-}1$ arecomplete blocks. (The empty message is treated as 1 incomplete block.)
2. If mn is a complete block then $mn' = k1 \oplus mn$ else $mn' = k2 \oplus (mn \parallel$ 10...02).3. Let $c0 = 00\ldots02$.
4. For $i = 1, ..., n\text{-}1$, calculate $ci = Ek(ci{-}1 \oplus mi)$.
5. $cn = Ek(cn{-}1 \oplus mn')$
6. Output $t = \text{msb}\ell(cn)$.

The verification process is as follows:

1. Use the above algorithm to generate the tag.
2. Check that the generated tag is equal to the received tag.

DIGITAL SIGNATURE

The most important development from the work on public-key cryptography is the digital signature. Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each

other. A digital signature is analogous to the handwritten signature, and provides a set of security capabilities that would be difficult to implement in any other way. It must have the following properties: • It must verify the author and the date and time of the signature

• It must to authenticate the contents at the time of the signature • It must be verifiableby third parties, to resolve disputes Thus, the digital signature function includes the authentication function. A variety of approaches has been proposed for the digitalsignature function. These approaches fall into two categories: direct and arbitrated.

• **Direct Digital Signature**

Direct Digital Signatures involve the direct application of public-key algorithms involvingonly the communicating parties. A digital signature may be formed by encrypting the entire message with the sender's private key, or by encrypting a hash code of the message with the sender's private key. Confidentiality can be provided by further encrypting the entire message plus signature using either public or private key schemes. It is important to perform the signature function first and then an outer confidentiality function, since in case of dispute, some third party must view the message and its signature. But these approaches are dependent on the security of the sender's private-key. Will have problemsif it is lost/stolen and signatures forged. Need time-stamps and timely key revocation.

Arbitrated Digital Signature

The problems associated with direct digital signatures can be addressed by using an arbiter, in a variety of possible arrangements. The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly. These schemes can be implemented with either private or public-key algorithms, and the arbiter may or may not see the actual message contents. **Using Conventional encryption**

$$X \rightarrow A : M \,||\, E (Kxa ,[IDx \,||\, H (M)])$$

$$A \rightarrow Y : E(Kay ,[IDx \,||\, M \,||\, E (Kxa ,[IDx \,||H(M))]) \,||\, T])$$

It is assumed that the sender X and the arbiter A share a secret key Kxa and that A and Y share secret key Kay. X constructs a message M and computes its hash value H(m) . Then X transmits the message plus a signature to A. the signature consists of an identifier IDx of X plus the hash value, all encrypted using Kxa.

A decrypts the signature and checks the hash value to validate the message. Then A transmits a message to Y, encrypted with Kay. The message includes IDx, the original message from X, the signature, and a timestamp.

Arbiter sees message

Problem : the arbiter could form an alliance with sender to deny a signed message, orwith the receiver to forge the sender's signature.

Using Public Key Encryption

$$X \to A : IDx \,||E(PRx,[\, IDx||\, E\,(PUy, E(PRx, M))])$$

$$A \to Y : E(PRa, [\, IDx \,||E\,(PUy, E\,(PRx, M))||\, T]\,)$$

X double encrypts a message M first with X's private key,PRx, and then with Y's public key, PUy. This is a signed, secret version of the message. This signed message, together with X's identifier , is encrypted again with PRx and, together with IDx, is sent to A. The inner, double encrypted message is secure from the arbiter (and everyone else except Y)

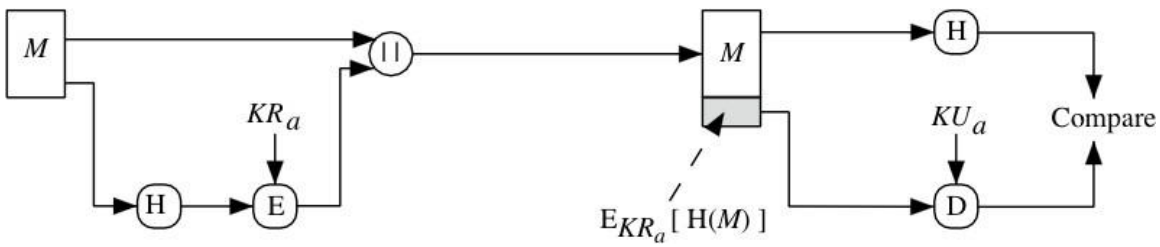
A can decrypt the outer encryption to assure that the message must have come from X (because only X has PRx). Then A transmits a message to Y, encrypted with PRa. The message includes IDx, the double encrypted message, and a timestamp.

Arbiter does not see message

Digital Signature Standard (DSS)

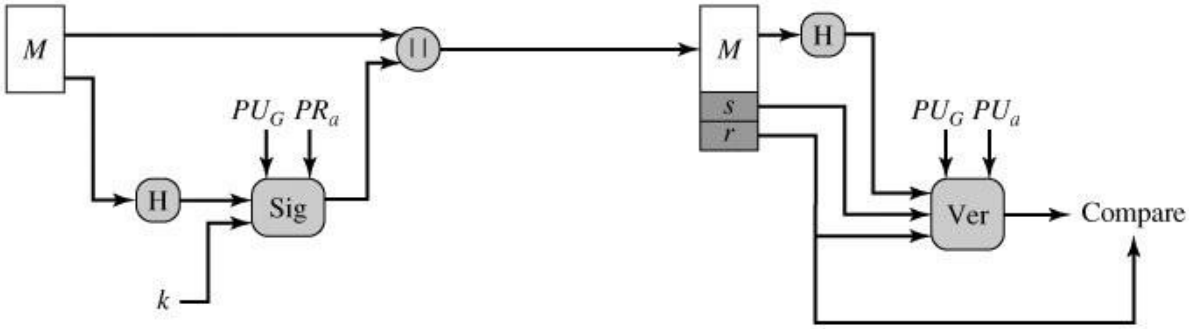
The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) and presents a new digitalsignature technique, the Digital Signature Algorithm (DSA). The DSS uses an algorithm that is designed to provide only the digital signature function and cannot be used for encryption or key exchange, unlike RSA.

The RSA approach is shown below. The message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted.



The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

The DSS approach also makes use of a hash function. 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).The result is a signature consisting of two components, labeled s and r.



(b) DSS approach

At the receiving end, the hash code of the incoming message is generated. This plus 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.

AUTHENTICATION APPLICATIONS

KERBEROS

Kerberos is an authentication service developed as part of Project Athena at MIT. It addresses the threats posed in an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. Some of these threats are:

- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from

the altered workstation appear to come from the impersonated workstation.

- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

Two versions of Kerberos are in current use: Version-4 and Version-5. The first published report on Kerberos listed the following requirements:

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

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

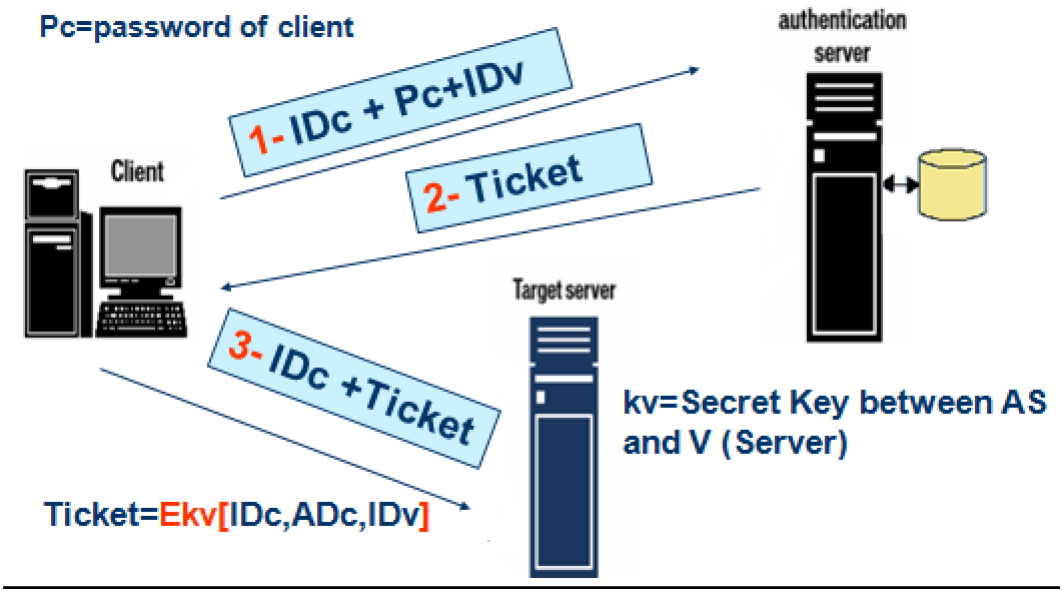
Transparent: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

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

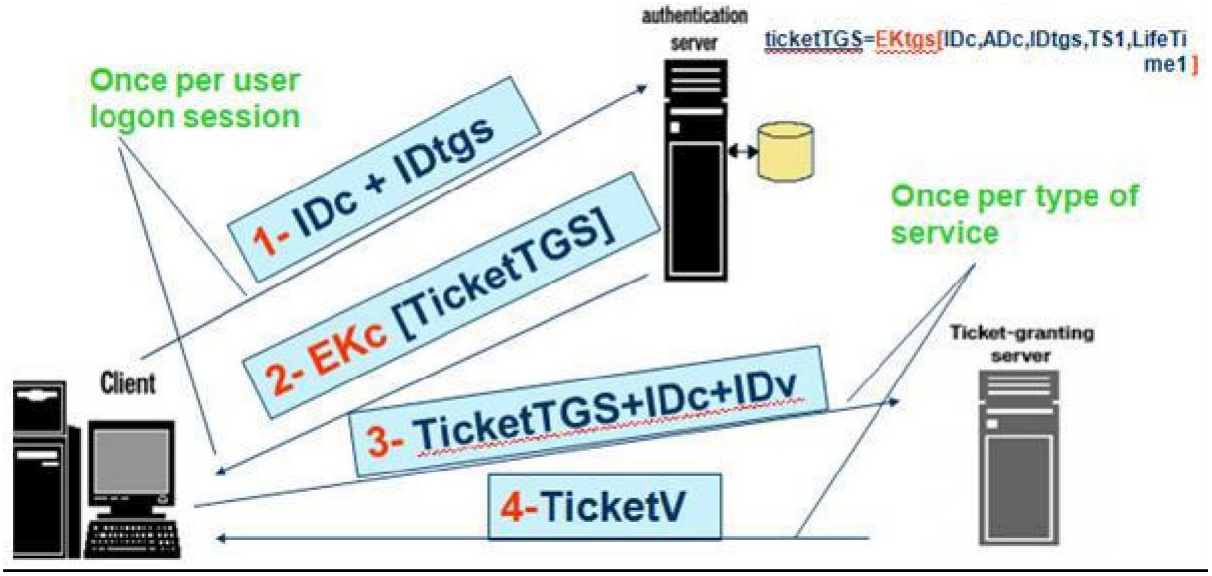
Two versions of Kerberos are in common use: Version 4 is most widely used version. Version 5 corrects some of the security deficiencies of Version 4. Version 5 has been issued as a draft Internet Standard (RFC 1510)

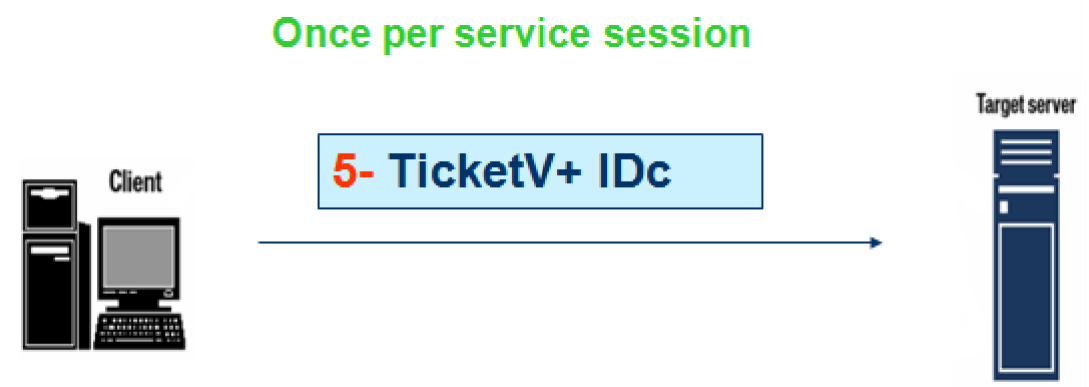
KERBEROS VERSION 4

1.) **SIMPLE DIALOGUE**:



MORE SECURE DIALOGUE





The Version 4 Authentication Dialogue The full Kerberos v4 authentication dialogue is shown here divided into 3 phases.

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

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

(3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

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

(5) C → V $Ticket_v \| Authenticator_c$

(6) V → C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

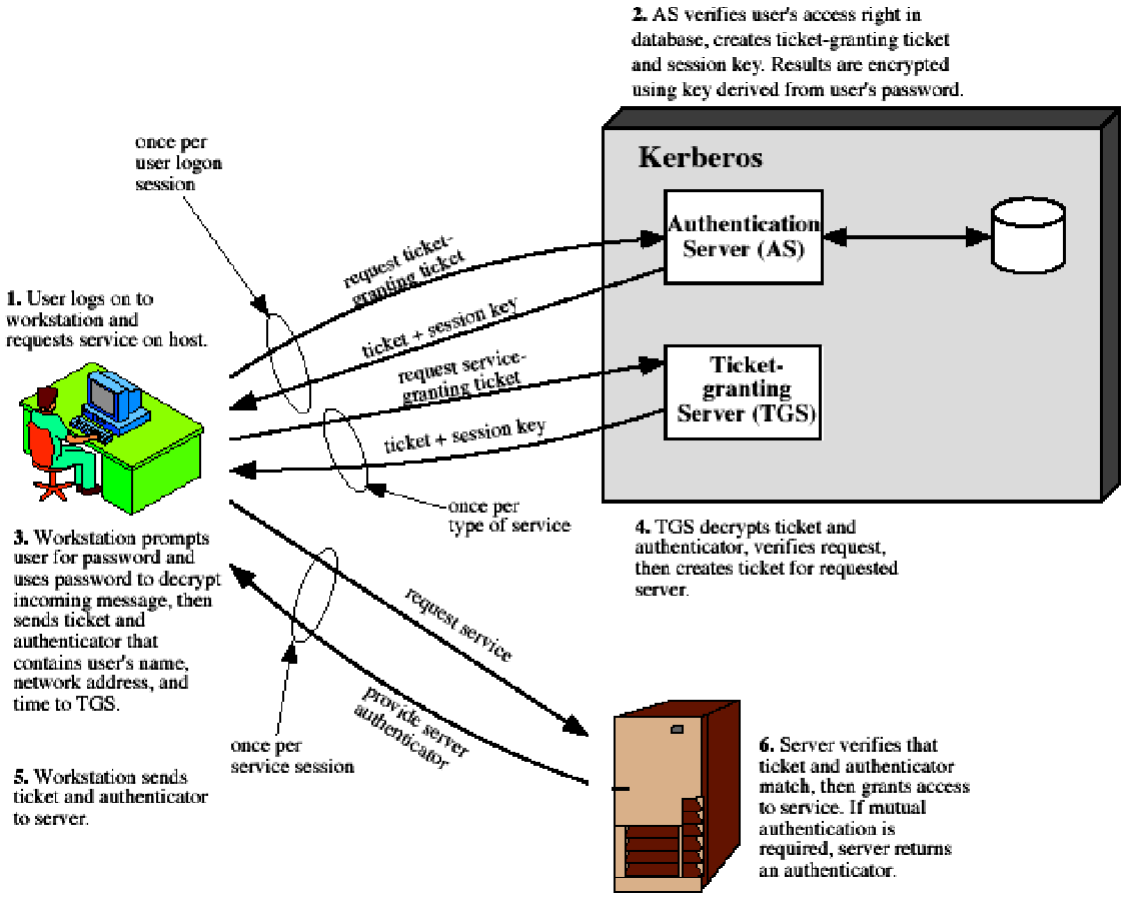
(c) Client/Server Authentication Exchange to obtain service

There is a problem of captured ticket-granting tickets and the need to determine that the ticket presenter is the same as the client for whom the ticket was issued. An efficient way of doing this is to use a session encryption key to secure information.

Message (1) includes a timestamp, so that the AS knows that the message is timely. Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the TGS and to learn its expiration time. Note that the ticket does not prove anyone's identity but is a way to distribute keys securely. It is the authenticator that proves the client's identity. Because the authenticator can be used only once and has a short lifetime, the threat of an opponent stealing both the ticket and the authenticator for presentation later is countered. C then sends the TGS a message that includes the ticket plus the ID of the requested service (message 3). The reply from the TGS, in message (4), follows the form of message (2). C now has a reusable service- granting ticket for V. When C presents this ticket, as shown in message (5), it also sends an authenticator.

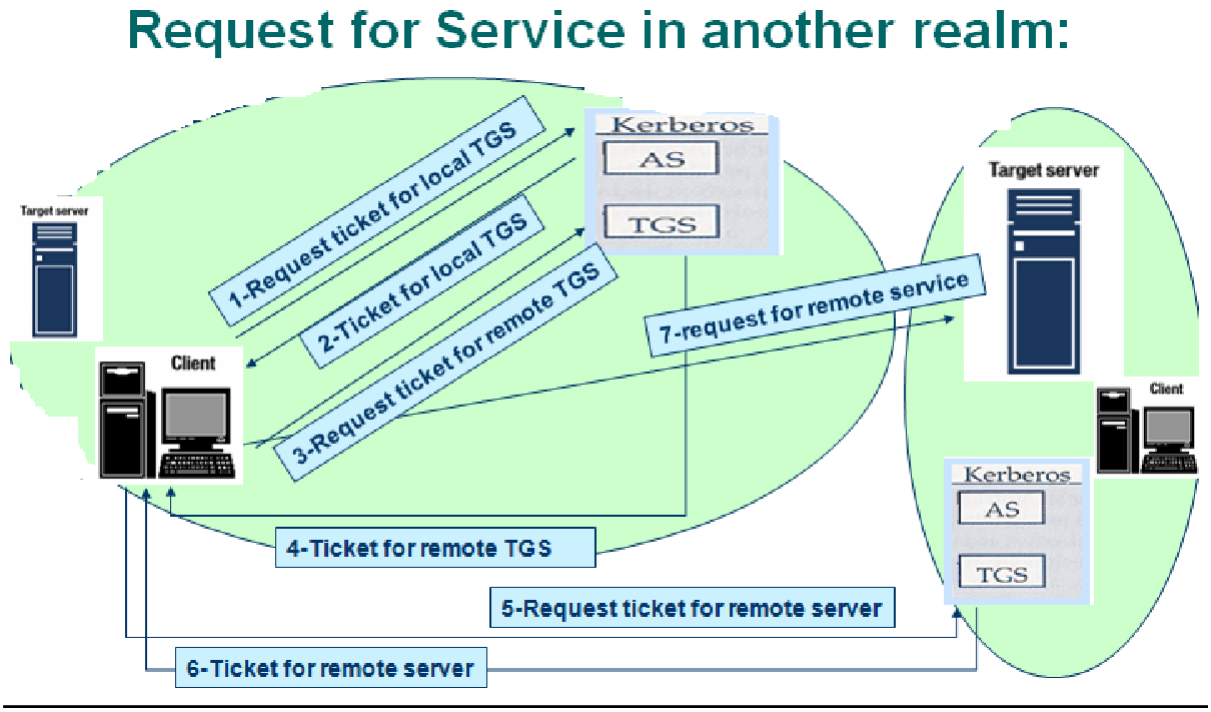
The server can decrypt the ticket, recover the session key, and decrypt the authenticator. If mutual authentication is required, the server can reply as shown in message (6).

Overview of Kerberos



Kerberos Realms A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers is referred to as a Kerberos realm. A Kerberos realm is a set of managed nodes that share the same Kerberos database, and are part of the same administrative domain. If have multiple realms, their Kerberos servers must share keys and trust each other.

The following figure shows the authentication messages where service is being requested from another domain. The ticket presented to the remote server indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request. One problem presented by the foregoing approach is that it does not scale well to many realms, as each pair of realms need to share a key.



The limitations of Kerberos version-4 are categorised into two types:

Environmental shortcomings of Version 4:

- Encryption system dependence: DES
- Internet protocol dependence
- Ticket lifetime
- Authentication forwarding
- Inter-realm authentication Technical

deficiencies of Version 4:

- Double encryption
- Session Keys
- Password attack

KERBEROS VERSION 5

Kerberos Version 5 is specified in RFC 1510 and provides a number of improvements over version 4 in the areas of environmental shortcomings and technical deficiencies. It includes some new elements such as:

Realm: Indicates realm of the user

Options

Times

– From: the desired start time for the ticket

– Till: the requested expiration time

– Rtime: requested renew-till time

Nonce: A random value to assure the response is fresh

The basic Kerberos version 5 authentication dialogue is shown here First, consider the **authentication service exchange.**

(1) C → AS Options || ID_c || $Realm_c$ || ID_{tgs} || $Times$ || $Nonce_1$

(2) AS → C $Realm_c$ || ID_C || $Ticket_{tgs}$ || E(K_c, [$K_{c,tgs}$ || $Times$ || $Nonce_1$ || $Realm_{tgs}$ || ID_{tgs}])

$Ticket_{tgs}$ = E(K_{tgs}, [$Flags$ || $K_{c,tgs}$ || $Realm_c$ || ID_C || AD_C || $Times$])

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

(3) C → TGS Options || ID_v || $Times$ || || $Nonce_2$ || $Ticket_{tgs}$ || $Authenticator_c$

(4) TGS → C $Realm_c$ || ID_C || $Ticket_v$ || E($K_{c,tgs}$, [$K_{c,v}$ || $Times$ || $Nonce_2$ || $Realm_v$ || ID_v])

$Ticket_{tgs}$ = E(K_{tgs}, [$Flags$ || $K_{c,tgs}$ || $Realm_c$ || ID_C || AD_C || $Times$])

$Ticket_v$ = E(K_v, [Flags || $K_{c,v}$ || $Realm_c$ || ID_C || AD_C || Times])

$Authenticator_c$ = E($K_{c,tgs}$, [ID_C || $Realm_c$ || TS_1])

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

(5) C → V Options || $Ticket_v$ || $Authenticator_c$

(6) V → C $E_{K_{C,V}}$ [TS_2 || Subkey || Seq#]

$Ticket_v$ = E(K_v, [Flags || $K_{c,v}$ || $Realm_c$ || ID_C || AD_C || Times])

$Authenticator_c$ = E($K_{c,v}$, [ID_C || $Realm_c$ || TS_2 || $Subkey$ || $Seq\#$])

(c) Client/Server Authentication Exchange to obtain service

Message (1) is a client request for a ticket-granting ticket. Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password. This block includes the session key to be used between the client and the TGS. Now compare the **ticket-granting service** exchange for versions 4 and 5. See that message (3) for both versions includes an authenticator, a ticket, and the name of the requested service. In addition, version 5 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1). The authenticator itself is essentially the same as the one used in version 4. Message (4) has the same structure as message (2), returning a ticket plus information needed by the client, the latter encrypted with the session key now shared by the client and the TGS. Finally, for the client/server authentication exchange, several new features appear in version 5, such as a request for mutual authentication. If required, the server responds with message (6) that includes the timestamp from the authenticator. The flags field included in tickets in version 5 supports expandedfunctionality compared to that available in version 4.

Advantages of Kerberos:

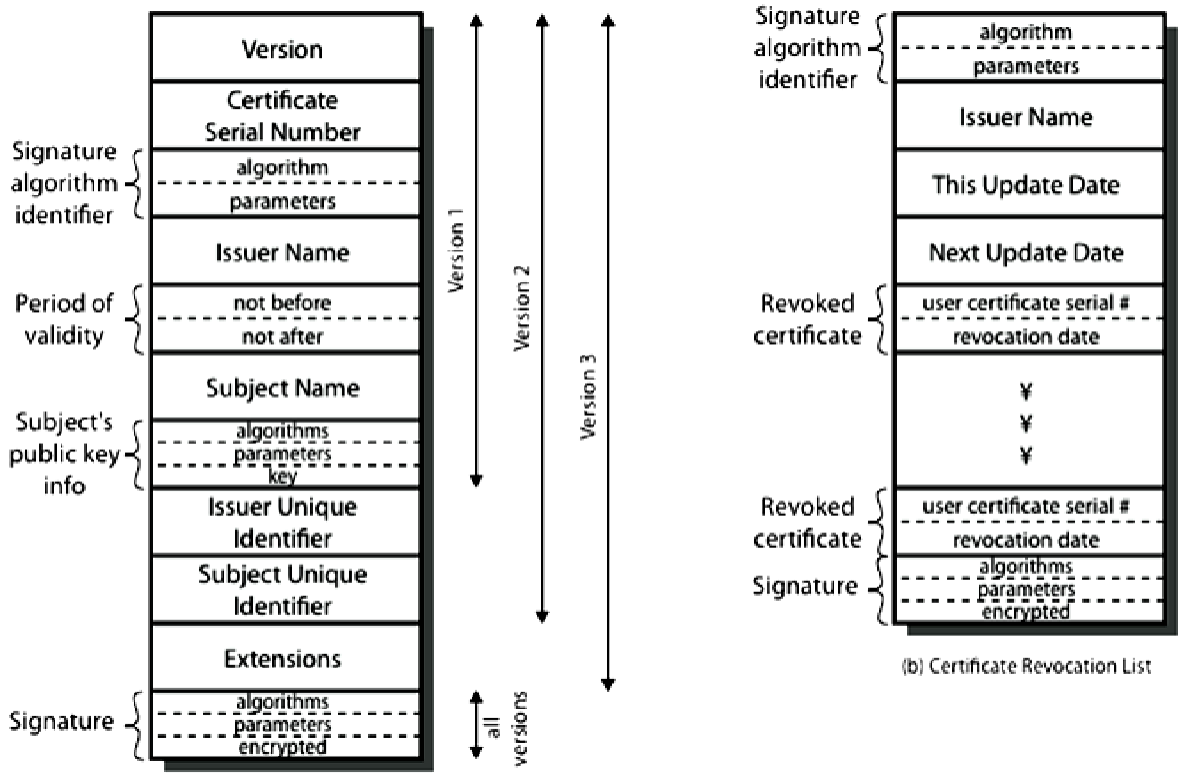
- User's passwords are never sent across the network, encrypted or in plain text
- Secret keys are *only* passed across the network in encrypted form
- Client and server systems mutually authenticate limits
- It the duration of their users' authentication.
- Authentications are **reusable** and **durable**
- Kerberos has been scrutinized by many of the top programmers, cryptologists andsecurity experts in the industry

X.509 AUTHENTICATION SERVICE

ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of serversthat 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 is based on the use of public-key cryptography and digital signatures. The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server 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.

The general format of a certificate is shown above, which includes the following elements:

- version 1, 2, or 3
- serial number (unique within CA) identifying certificate
- signature algorithm identifier
- issuer X.500 name (CA)
- period of validity (from - to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2+)



subject unique identifier (v2+)

extension fields (v3)

signature (of hash of all fields in certificate)

The standard uses the following notation to define a certificate:

CA<<A>> = CA {V, SN, AI, CA, TA, A, Ap}

Where Y<<X>>= the certificate of user X issued by certification authority Y

Y {I} == the signing of I by Y. It consists of I with an encrypted hash code appended

User certificates generated by a CA have the following characteristics:

Any user with CA's public key can verify the user public key that was certifiedNo

party other than the CA can modify the certificate without being detected because

they cannot be forged, certificates can be placed in a public directory

Scenario: Obtaining a User Certificate If both users share a common CA then they are assumed to know its public key. Otherwise CA's must form a hierarchy and use certificates linking members of hierarchy to validate other CA's. Each CA has certificates for clients (forward) and parent (backward). Each client trusts parents' certificates. It

enables verification of any certificate from one CA by users of all other CAs in hierarchy. A has obtained a certificate from the CA X1. B has obtained a certificate from the CA X2. Acan read the B's certificate but cannot verify it. In order to solve the problem ,the Solution: **X1<<X2> X2<<B>>.** A obtain the certificate of X2 signed by X1 from directory. obtain X2's public key. A goes back to directory and obtain the certificate of B signed by X2. obtain B's public key securely. 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

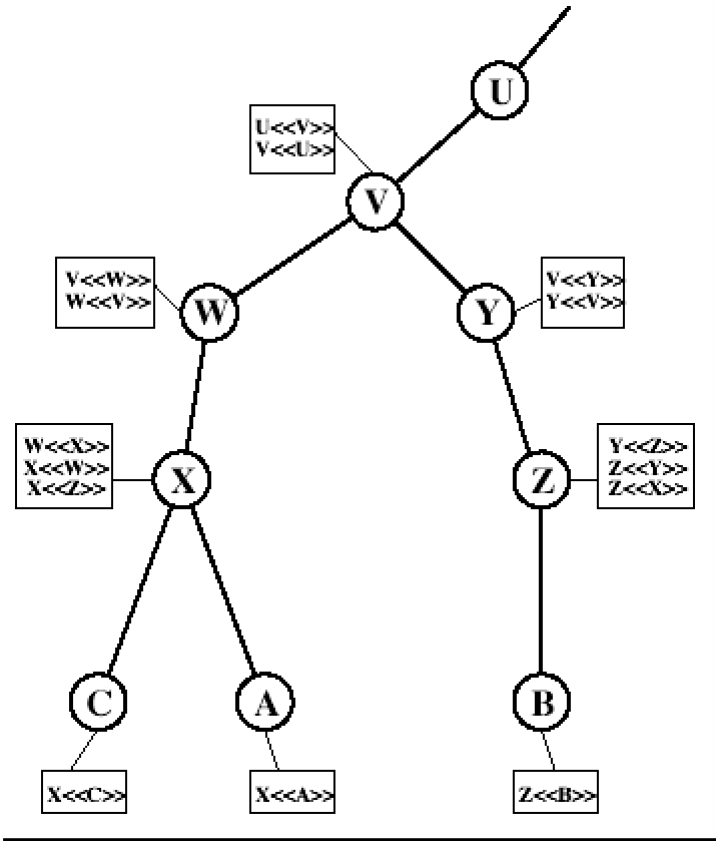
X.509 CA Hierarchy

A acquires B certificate using chain:

X<<W>>W<<V>>V<<Y>>Y<<Z>> Z<<B>>

B acquires A certificate using chain:

Z<<Y>>Y<<V>>V<<W>>W<<X>> X<<A>>



Revocation of Certificates Typically, a new certificate is issued just 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:

The user's private key is assumed to be compromised.

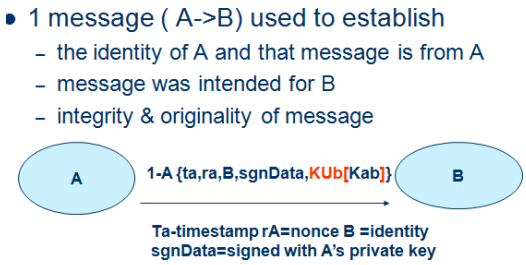
The user is no longer certified by this CA.

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 also be posted on the directory. Each **certificate revocation list (CRL) posted** to the directory is signed by the issuer and includes 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.

AUTHENTICATION PROCEDURES

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications. All these procedures make use of public-key signatures. It is assumed that the two parties know each other's public key, either by obtaining each other's certificates from the directory or because the certificate is included in the initial message from each side. 1. One-Way Authentication: One way authentication involves a single transfer of information from one user (A) to another (B), and establishes the details shown above. Note that only the identity of the initiating entity is verified in this process, not that of the responding entity. At a minimum, the message includes a timestamp ,a nonce, and the identity of B and is signed with A's private key. The message may also include information to be conveyed, such as a session key for B.

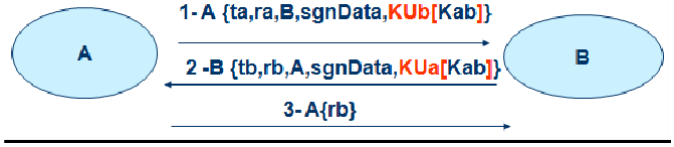


Two-Way Authentication: Two-way authentication thus permits both parties in a communication to verify the identity of the other, thus additionally establishing the above details. The reply message includes the nonce from A, to validate the reply. It also includes a timestamp and nonce generated by B, and possible additional information for A.

- 2 messages (A->B, B->A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply

Three-Way Authentication: Three-Way Authentication includes a final message from A to B, which contains a signed copy of the nonce, so that timestamps need not be checked, for use when synchronized clocks are not available.

- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks



UNIT-4

***Email Privacy**: Pretty Good Privacy (PGP) and S/MIME. **IP Security:** IP Security Overview, IP Security Architecture, Authentication Header, Encapsulating Security Payload, Combining Security Associations and Key Management.*

PRETTY GOOD PRIVACY

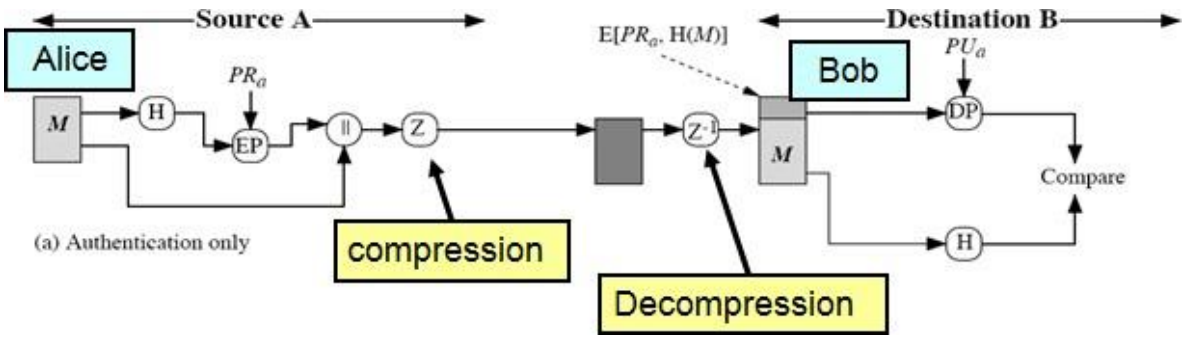
In virtually all distributed environments, electronic mail is the most heavily used network-based application. But current email services are roughly like "postcards", anyone who wants could pick it up and have a look as it's in transit or sitting in the recipients mailbox. PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. With the explosively growing reliance on electronic mail for every conceivable purpose, there grows a demand for authentication and confidentiality services. The Pretty Good Privacy (PGP) secure email program, is a remarkable phenomenon, has grown explosively and is now widely used. Largely the effort of a single person, Phil Zimmermann, who selected the best available crypto algorithms to use & integrated them into a single program, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. It is independent of government organizations and runs on a wide range of systems, in both free & commercial versions. *There are **five** important services inPGP*

- *Authentication (Sign/Verify)*
- *Confidentiality (Encryption/Decryption)*
- *Compression*
- *Email compatibility*
- *Segmentation and Reassembly*
- The last three are **transparent** to the user

PGP Notations:

Ks	=session key used in symmetric encryption scheme
PRa	=private key of user A, used in public-key encryption scheme
PUa	=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

PGP Operation- Authentication



1. sender creates message

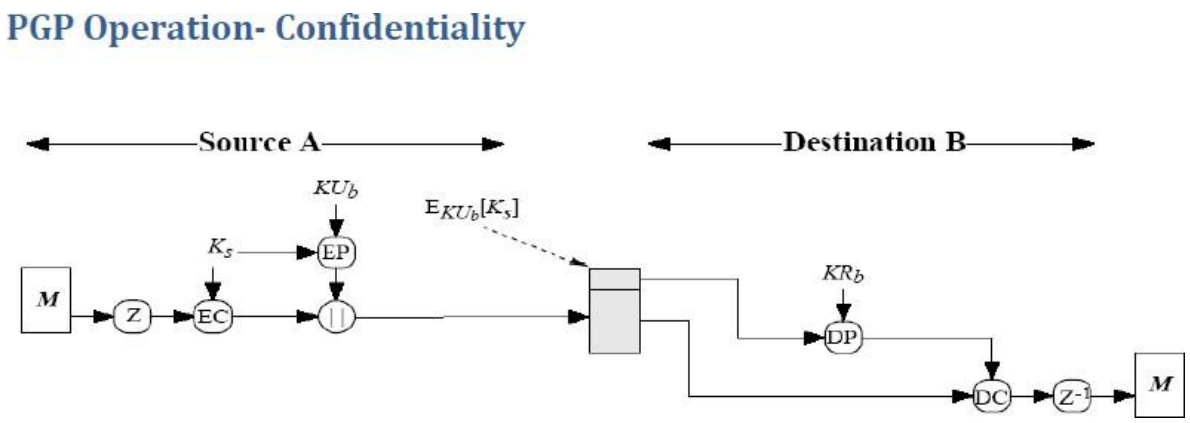
2. use SHA-1 to generate 160-bit hash of message

3. signed hash with RSA using sender's private key, and is attached to message

4. receiver uses RSA with sender's public key to decrypt and recover hash code

5. receiver verifies received message using hash of it and compares with decrypted hashcode

PGP Operation- Confidentiality



Sender:

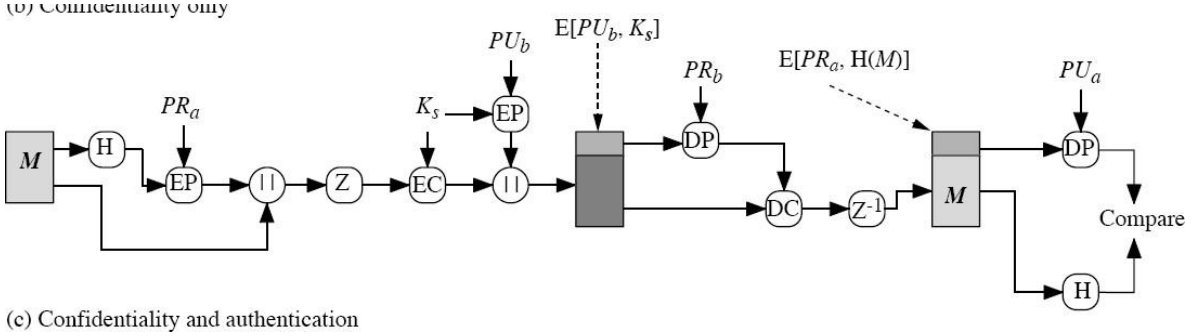
1. Generates message and a random number (session key) only for this message
2. Encrypts message with the session key using AES, 3DES, IDEA or CAST-128
3. Encrypts session key itself with recipient's public key using RSA
4. Attaches it to message

Receiver:

1. Recovers session key by decrypting using his private key
2. Decrypts message using the session key

Confidentiality service provides no assurance to the receiver as to the identity of sender (i.e. no authentication). Only provides confidentiality for sender that only the recipient can read the message (and no one else)

PGP Operation – Confidentiality & Authentication



can use both services on same message o create signature & attach to message o encrypt both message & signature

o attach RSA/ElGamal encrypted session key

o *is called* ***authenticated confidentiality***

PGP Operation – Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage. The placement of the compression algorithm, indicated by Z for compression and Z-1 for decompression is critical. The compression algorithm used is ZIP.

The signature is generated before compression for two reasons:

1. so that one can store only the uncompressed message together with signature for later verification

2. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm as the PGP compression algorithm is not deterministic

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.

PGP Operation – Email Compatibility

When PGP is used, at least part of the block to be transmitted is encrypted, and thus consists of a stream of arbitrary 8-bit octets. However many electronic mail systems only permit the use of ASCII text. To accommodate this restriction, PGP provides the service

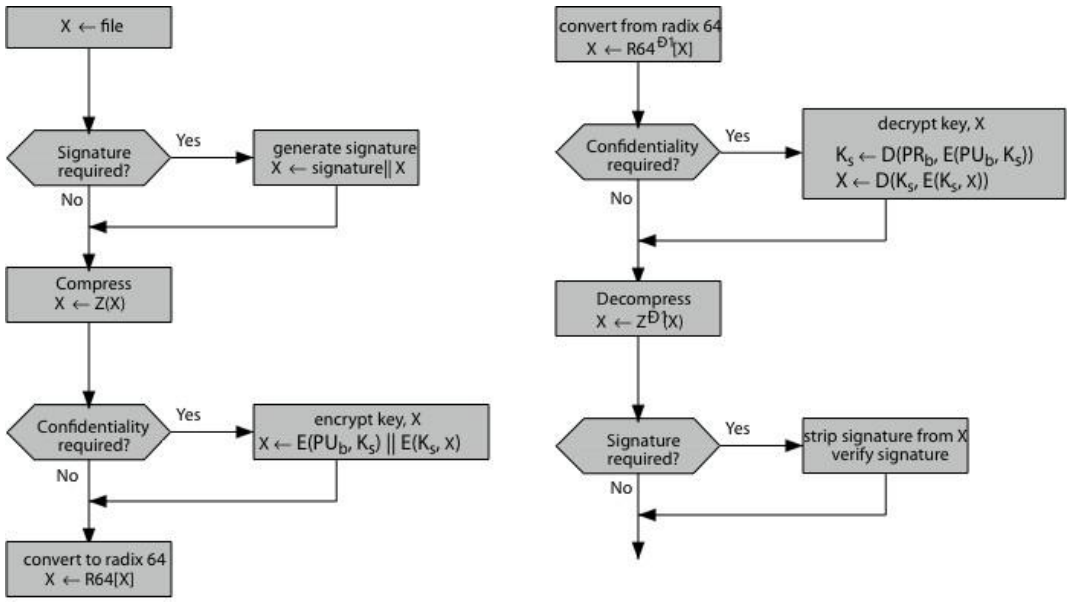
of converting the raw 8-bit binary stream to a stream of printable ASCII characters. It uses radix-64 conversion, in which each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors. The use of radix 64 expands a message by 33%, but still an overall compression of about one-third can be achieved.

PGP Operation - Segmentation/Reassembly

E-mail facilities often are restricted to a maximum message length. For example, many of the facilities accessible through the Internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately. To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation is done after all of the other processing, including the radix-64 conversion. Thus, the session key component and signature component appear only once, at the beginning of the first segment. Reassembly at the receiving end is required before verifying signature or decryption

PGP Operations – Summary

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key, and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key, and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix 64 conversion.
Segmentation	—	To accommodate maximum message size limitations, PGP performs segmentation and reassembly.



(a) Generic Transmission Diagram (from A)

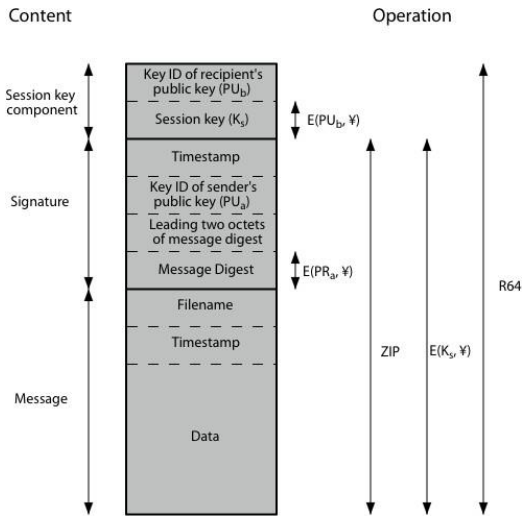
(b) Generic Reception Diagram (to B)

PGP Message Format

A message consists of three components: the message component, a signature (optional), and a session key component (optional). The *message component* includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation. The *signature component* includes the following:

- *Timestamp*: The time at which the signature was made.
- *Message digest*: The 160-bit SHA-1 digest, encrypted with the sender's private signature key.
- *Leading two octets of message digest*: To enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication, by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.

Key ID of sender's public key: Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest



Notation:

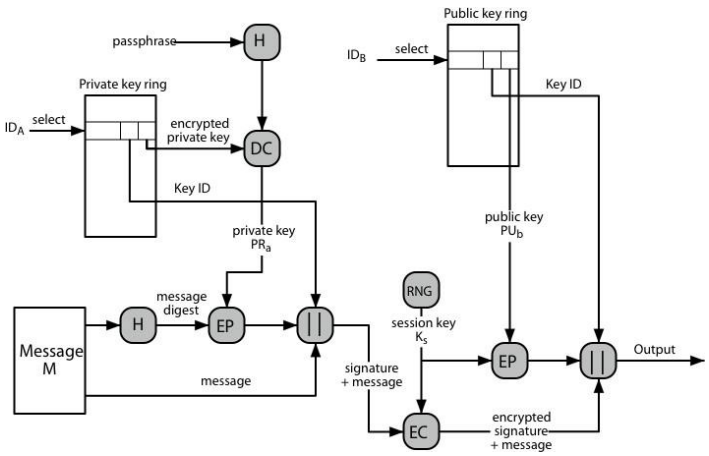
$E(PU_b, \bullet)$ = encryption with user b's public key
$E(PR_a, \bullet)$ = encryption with user a's private key
$E(K_s, \bullet)$ = encryption with session key
ZIP = Zip compression function
R64 = Radix-64 conversion function

The *session key component* includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key. The entire block is usually encoded with radix-64 encoding.

PGP Message Transmission and Reception

Message transmission

The following figure shows the steps during message transmission assuming that the message is to be both signed and encrypted.



The sending PGP entity performs the following steps:

Signing the message

a. PGP retrieves the sender's private key from the private-key ring using your_userid as an index. If your_userid was not provided in the command, the first private key on the ring is retrieved.

b. PGP prompts the user for the passphrase to recover the unencrypted private key.

c. The signature component of the message is constructed

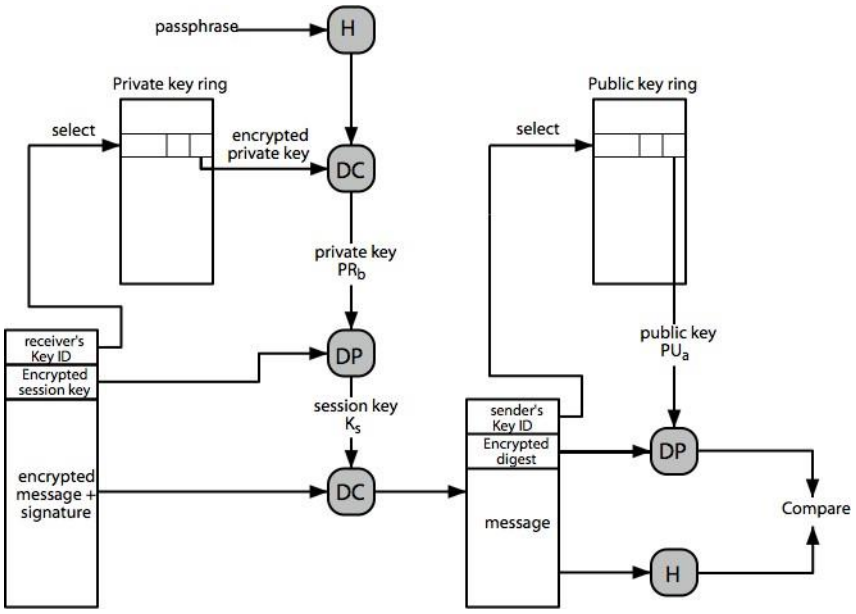
Encrypting the message

a. PGP generates a session key and encrypts the message.

b. PGP retrieves the recipient's public key from the public-key ring using her_userid as an index.

c. The session key component of the message is constructed.

Message Reception



The receiving PGP entity performs the following steps:

Decrypting the message

a. PGP retrieves the receiver's private key from the private-key ring, using the Key ID fieldin the session key component of the message as an index.

b. PGP prompts the user for the passphrase to recover the unencrypted private key.

c. PGP then recovers the session key and decrypts the message.

Authenticating the message

a. PGP retrieves the sender's public key from the public-key ring, using the Key ID field inthe signature key component of the message as an index.

b. PGP recovers the transmitted message digest.

c. PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, which in turn provided support for varying content types and multi-part messages over the text only support in the original InternetRFC822 email standard. MIME allows encoding of binary data to textual form for transport over traditional RFC822 email systems. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851 and S/MIME support is now included in many modern mail agents.

RFC 822

RFC 822 defines a format for text messages that are sent using electronic mail and it has been the standard for Internet-based text mail message. The overall structure of a message that conforms to RFC 822 is very simple. A message consists of some number ofheader lines (the header) followed by unrestricted text (the body). The header is separated from the body by a blank line. A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords are *From*, *To*, *Subject*, and *Date*.

Multipurpose Internet Mail Extensions

MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail. **Problems with RFC 822 and SMTP**

• Executable files or other binary objects must be converted into ASCII. Various schemes exist (e.g., Unix UUencode), but a standard is needed

• Text data that includes special characters (e.g., Hungarian text) cannot be transmittedas SMTP is limited to 7-bit ASCII

• Some servers reject mail messages over a certain size

• Some common problems exist with the SMTP implementations which do not adhere completely to the SMTP standards defined in RFC 821. They are:

- delete, add, or reorder CR and LF characters
- truncate or wrap lines longer than 76 characters
- remove trailing white space (tabs and spaces) pad
- lines in a message to the same length convert tab
- characters into multiple spaces

MIME is intended to resolve these problems in a manner that is compatible with existingRFC 822 implementations and the specification is provided in RFC's 2045 through 2049.

The MIME specification includes the following elements:

1. Five new message header fields are defined, which provide information about the bodyof the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that protect the content from alteration by the mail system.

MIME - New header fields The five header fields defined in MIME are as follows:

• *MIME-Version:* Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.

• *Content-Type*: Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.

• *Content-Transfer-Encoding*: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.

• Content-ID: Used to identify MIME entities uniquely in multiple contexts.

• *Content-Description*: A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Content Types The bulk of the MIME specification is concerned with the definition of a variety of content types. There are seven different major types of content and a total of 15 subtypes. In general, a content type declares the general type of data, and the subtype specifies a particular format for that type of data. For the text type of body, the primary subtype is plain text, which is simply a string of ASCII characters or ISO 8859 characters. The enriched subtype allows greater formatting flexibility. The multipart type indicates that the body contains multiple, independent parts. The Content-Type header field includes a parameter called boundary that defines the delimiter between body parts. This boundary should not appear in any parts of the message. Each boundary starts on a new line and consists of two hyphens followed by the boundary value. The final boundary, which indicates the end of the last part, also has a suffix of two hyphens. Within each part, there may be an optional ordinary MIME header. There are four subtypes of the multipart type, all of which have the same overall syntax.

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.
	octet-stream	General binary data consisting of 8-bit bytes.

The message type provides a number of important capabilities in MIME. The message/rfc822 subtype indicates that the body is an entire message, including header

and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 822 message, but also any MIME message. The message/partial subtype enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this subtype, three parameters are specified in the Content-Type: Message/Partial field: an id common to all fragments of the same message, a sequence number unique to each fragment, and the total number of fragments. The message/external-body subtype indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. The application type refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-based application.

MIME Transfer Encodings The other major component of the MIME specification, in addition to content type specification, is a definition of transfer encodings for message bodies. The objective is to provide reliable delivery across the largest range of environments.

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 necessarily 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 MIME standard defines two methods of encoding data. The Content-Transfer- Encoding field can actually take on six values. Three of these values (7bit, 8bit, and binary) indicate that no encoding has been done but provide some information about the nature of the data. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used, for which a name is to be supplied. The two actual encoding schemes defined are quoted-printable and base64. Two schemes are defined to provide a choice between a transfer technique that is essentially human

readable and one that is safe for all types of data in a way that is reasonably compact. The quoted-printable transfer encoding is useful when the data consists largely of octets that correspond to printable ASCII characters. In essence, it represents nonsafe characters by the hexadecimal representation of their code and introduces reversible (soft) line breaks to limit message lines to 76 characters. The base64 transfer encoding, also known as radix-64 encoding, is a common one for encoding arbitrary binary data in such a way as to be invulnerable to the processing by mail transport programs.

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.

Native Form	The body to be transmitted is created in the system's native format. The native character set is used and, where appropriate, local end-of-line conventions are used as well. The body may be a UNIX-style text file, or a Sun raster image, or a VMS indexed file, or audio data in a system-dependent format stored only in memory, or anything else that corresponds to the local model for the representation of some form of information. Fundamentally, the data is created in the "native" form that corresponds to the type specified by the media type.
Canonical Form	The entire body, including "out-of-band" information such as record lengths and possibly file attribute information, is converted to a universal canonical form. The specific media type of the body as well as its associated attributes dictate the nature of the canonical form that is used. Conversion to the proper canonical form may involve character set conversion, transformation of audio data, compression, or various other operations specific to the various media types. If character set conversion is involved, however, care must be taken to understand the semanties of the media type, which may have strong implications for any character set conversion (e.g. with regard to syntactically meaningful characters in a text subtype other than "plain").

S/MIME Functionality

S/MIME has a very similar functionality to PGP. Both offer the ability to sign and/or encrypt messages.

Functions

S/MIME provides the following functions:

• **Enveloped data**: This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.

• **Signed data**: A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.

• **Clear-signed data**: As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.

• **Signed and enveloped data**: Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

IP SECURITY OVERVIEW

Definition: Internet Protocol security (IPSec) is a framework of open standards for protecting communications over Internet Protocol (IP) networks through the use of cryptographic security services. IPSec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

Need for IPSec

In Computer Emergency Response Team (CERT)'s 2001 annual report it listed 52,000 security incidents in which most serious types of attacks included **IP spoofing**, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP and various forms of **eavesdropping and packet sniffing**, inwhich attackers read transmitted information, including logon information and database contents. In response to these issues, the IAB included authentication and encryption as necessary security features in the next-generation IP i.e. IPv6.

Applications of IPSec

IPSec provides the capability to secure communications across a LAN, across private and

public wide area networks (WAN's), and across the Internet.

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

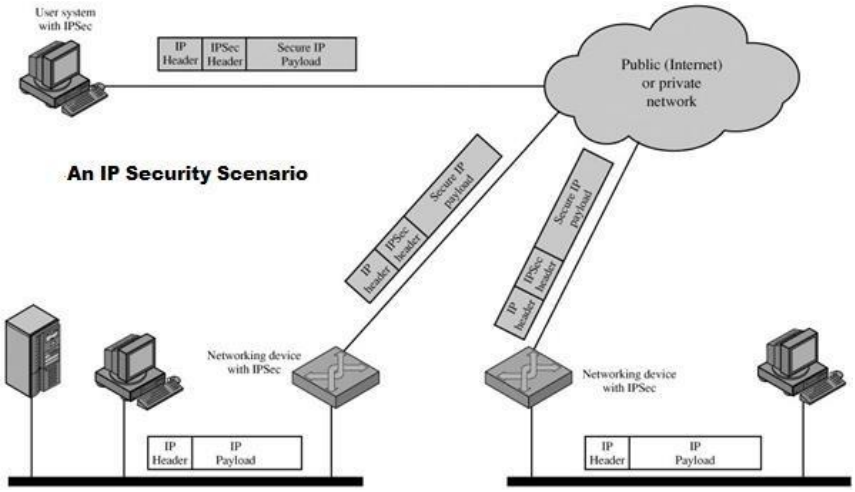
• ***Secure remote access over the Internet***: An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for travelling employees and telecommuters.

• ***Establishing extranet and intranet connectivity with partners***: IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

• ***Enhancing electronic commerce security***: Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

The principal feature of IPSec enabling it to support varied applications is that it can encrypt and/or authenticate all traffic at IP level. Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can besecured.

The following figure shows a typical scenario of IPSec usage. An organization maintains LANs at dispersed locations. Non secure IP traffic is conducted on each LAN.



The IPSec protocols operate in networking devices, such as a router or firewall that connect each LAN to the outside world. The IPSec networking device will typically encrypt and compress all traffic going into the WAN, and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocols to provide security.

Benefits of IPSec

The benefits of IPSec are listed below:

• IPSec in a firewall/router provides strong security to all traffic crossing the perimeter

• IPSec in a firewall is resistant to bypass

• IPSec is below transport layer(TCP,UDP), hence transparent to applications

• IPSec can be transparent to end users

• IPSec can provide security for individual users if needed (useful for offsite workers and setting up a secure virtual subnetwork for sensitive applications)

Routing Applications

IPSec also plays a vital role in the routing architecture required for internetworking. It assures that:

• router advertisements come from authorized routers

• neighbor advertisements come from authorized routers

• redirect messages come from the router to which initial packet was sent

• A routing update is not forged

IP SECURITY ARCHITECTURE

To understand IP Security architecture, we examine IPSec documents first and then moveon to IPSec services and Security Associations.

IPSec Documents

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

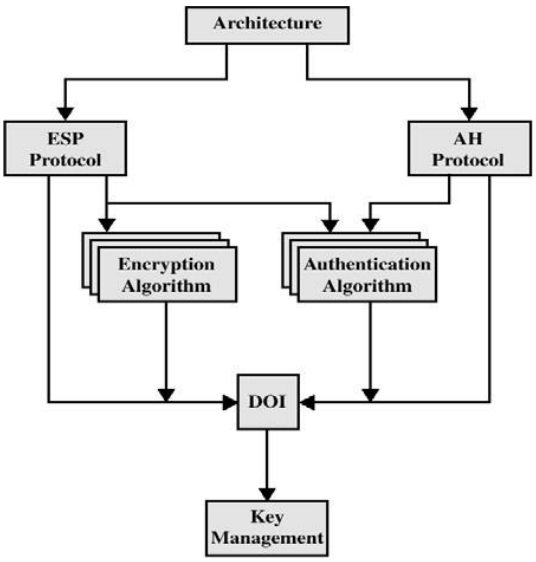
• RFC 2401: An overview of a security architecture

• RFC 2402: Description of a packet authentication extension to IPv4 and IPv6

• RFC 2406: Description of a packet encryption extension to IPv4 and IPv6

• RFC 2408: Specification of key management capabilities

Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header. The extension header for authentication is known as the Authentication header; that for encryption is known as the Encapsulating Security Payload (ESP) header. In addition to these four RFCs, a number of additional drafts have been published by the IP Security Protocol Working Group set up by the IETF. The documents are divided into seven groups, as depicted in following figure:



• **Architecture**: Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology

• **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.

• **Authentication Header (AH):** Covers the packet format and general issues related tothe use of AH for packet authentication.

• **Encryption Algorithm**: A set of documents that describe how various encryption algorithms are used for ESP.

• **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.

• **Key Management**: Documents that describe key management schemes.

• **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

IPSec Services

IPSec architecture makes use of two major protocols (i.e., Authentication Header and ESP protocols) for providing security at IP level. This facilitates the system to beforehand choose an algorithm to be implemented, security protocols needed and any cryptographickeys required to provide requested services. The IPSec services are as follows:

Connectionless Integrity:- Data integrity service is provided by IPSec via AH which prevents the data from being altered during transmission.

Data Origin Authentication:- This IPSec service prevents the occurrence of replay attacks, address spoofing etc., which can be fatal

Access Control:- The cryptographic keys are distributed and the traffic flow is controlled in both AH and ESP protocols, which is done to accomplish access control over the data transmission.

Confidentiality:- Confidentiality on the data packet is obtained by using an encryption technique in which all the data packets are transformed into ciphertext packets which are unreadable and difficult to understand.

Limited Traffic Flow Confidentiality:- This facility or service provided by IPSec ensures that the confidentiality is maintained on the number of packets transferred or received. This can be done using padding in ESP.

Replay packets Rejection:- The duplicate or replay packets are identified and discarded using the sequence number field in both AH and ESP.

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

SECURITY ASSOCIATIONS

Since IPSEC is designed to be able to use various security protocols, it uses Security Associations (SA) to specify the protocols to be used. SA is a database record which specifies security parameters controlling security operations. They are referenced by the sending host and established by the receiving host. An index parameter called the Security Parameters Index (SPI) is used. SAs are in one direction only and a second SA must be established for the transmission to be bi-directional. A security association is uniquely identified by three parameters:

• **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving systemto select the SA under which a received packet will be processed.

• **IP Destination Address**: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.

• **Security Protocol Identifier**: This indicates whether the association is an AH or ESP security association.

SA Parameters

In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:

• **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers

• **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations).

• **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay

• **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations).

• **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).

• **Lifetime of This Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementations).

• **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section.

• **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).

Transport and Tunnel Modes

Both AH and ESP support two modes of use: transport and tunnel mode.

	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 plus selected portions of outer IP header
ESP	**Encrypts** IP payload and any IPv6 extesion header	**Encrypts** inner IP packet
ESP with authentication	**Encrypts** IP payload and any IPv6 extesion header. **Authenticates** IP payload but no IP header	**Encrypts** inner IP packet. **Authenticates** inner IP packet

IP sec can be used (both AH packets and ESP packets) in two modes

• **Transport mode**: the IP sec header is inserted just after the IP header –this containsthe security information, such as SA identifier, encryption, authentication

Typically used in end-to-end communicationIP

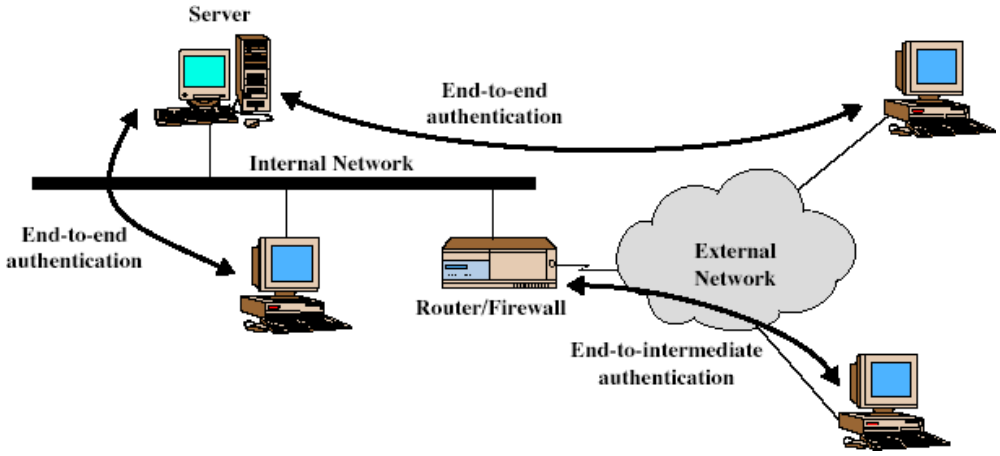
header not protected

• **Tunnel mode**: the entire IP packet, header and all, is encapsulated in the body of a newIP packet with a completely new IP header

Typically used in firewall-to-firewall communication

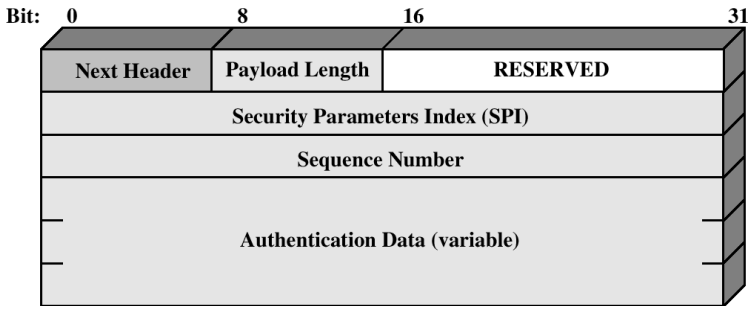
Provides protection for the whole IP packet

No routers along the way will be able (and will not need) to check the content of the packets



AUTHENTICATION HEADER

The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed in today's Internet. The AH also guards against the replay attack. Authentication is based on the use of a message authentication code (MAC), hence the two parties must share a secret key. The Authentication Header consists of the following fields:



IPSec Authentication Header

• **Next Header (8 bits):** Identifies the type of header immediately following this header.

• **Payload Length (8 bits):** Length of Authentication Header in 32-bit words, minus 2. For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header, and the Payload Length field has a value of 4.

• **Reserved (16 bits):** For future use.

• **Security Parameters Index (32 bits):** Identifies a security association.

• **Sequence Number (32 bits):** A monotonically increasing counter value, discussedlater.

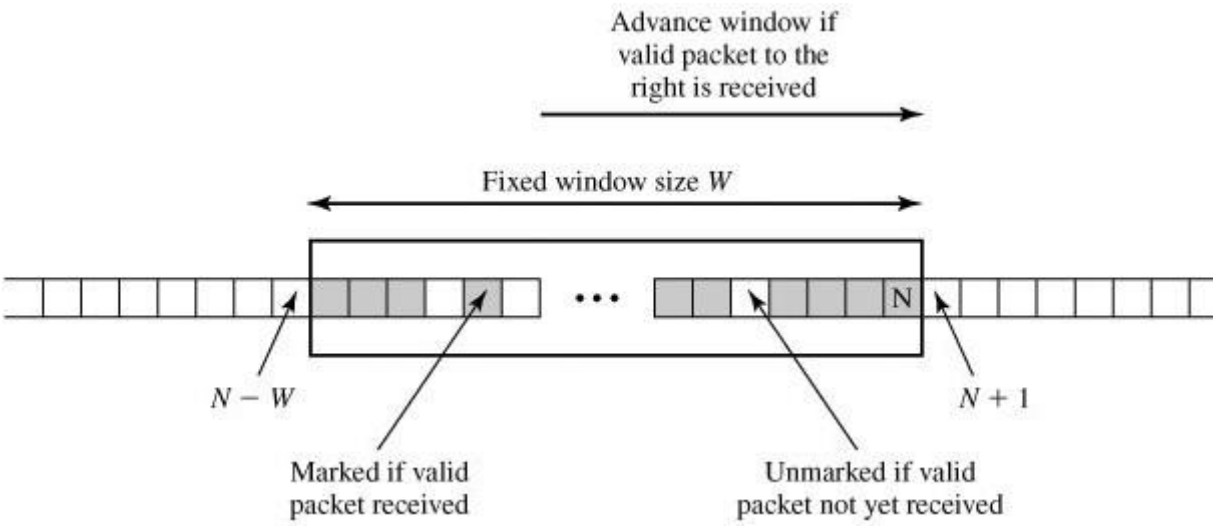
• **Authentication Data (variable):** A variable-length field (must be an integral numberof 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.

Anti-Replay Service

Anti-replay service is designed to overcome the problems faced due to replay attacks in which an intruder intervenes the packet being transferred, make one or more duplicate copies of that authenticated packet and then sends the packets to the desired destination, thereby causing inconvenient processing at the destination node. The Sequence Number field is designed to thwart such attacks.

When a new SA is established, the sender initializes a sequence number counter to 0. Eachtime that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. This value goes on increasing with respect to the number of packets being transmitted. The sequence number field in each packet represents the value of this counter. The maximum value of the sequence number field can go up to 232-1. If the limit of 232-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key.

The IPSec authentication document dictates that the receiver should implement a window of size W, with a default of W = 64. The right edge of the window represents the highest sequence number, N, so far received for a valid packet. For any packet with a sequence number in the range from N-W+1 to N that has been correctly received (i.e., properly authenticated), the corresponding slot in the window is marked as shown. Inbound processing proceeds as follows when a packet is received:



Antireplay Mechanism

1. If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.
2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.
3. If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

Integrity Check Value

ICV is the value present in the authenticated data field of ESP/AH, which is used to determine any undesired modifications made to the data during its transit. ICV can also be referred as MAC or part of MAC algorithm. MD5 hash code and SHA-1 hash code are implemented along with HMAC algorithms i.e.,

• HMAC-MD5-96

• HMAC-SHA-1-96

In both cases, the full HMAC value is calculated but then truncated by using the first 96 bits, which is the default length for the Authentication Data field. The MAC is calculated over

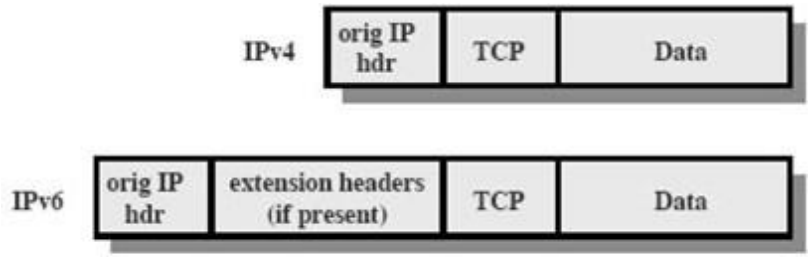
• IP header fields that either do not change in transit (immutable) or that are predictable in value upon arrival at the endpoint for the AH SA. Fields that may change in transit and whose value on arrival is unpredictable are set to zero for purposes of calculation at both source and destination.

• The AH header other than the Authentication Data field. The Authentication Data field is set to zero for purposes of calculation at both source and destination.

• The entire upper-level protocol data, which is assumed to be immutable in transit (e.g., a TCP segment or an inner IP packet in tunnel mode).

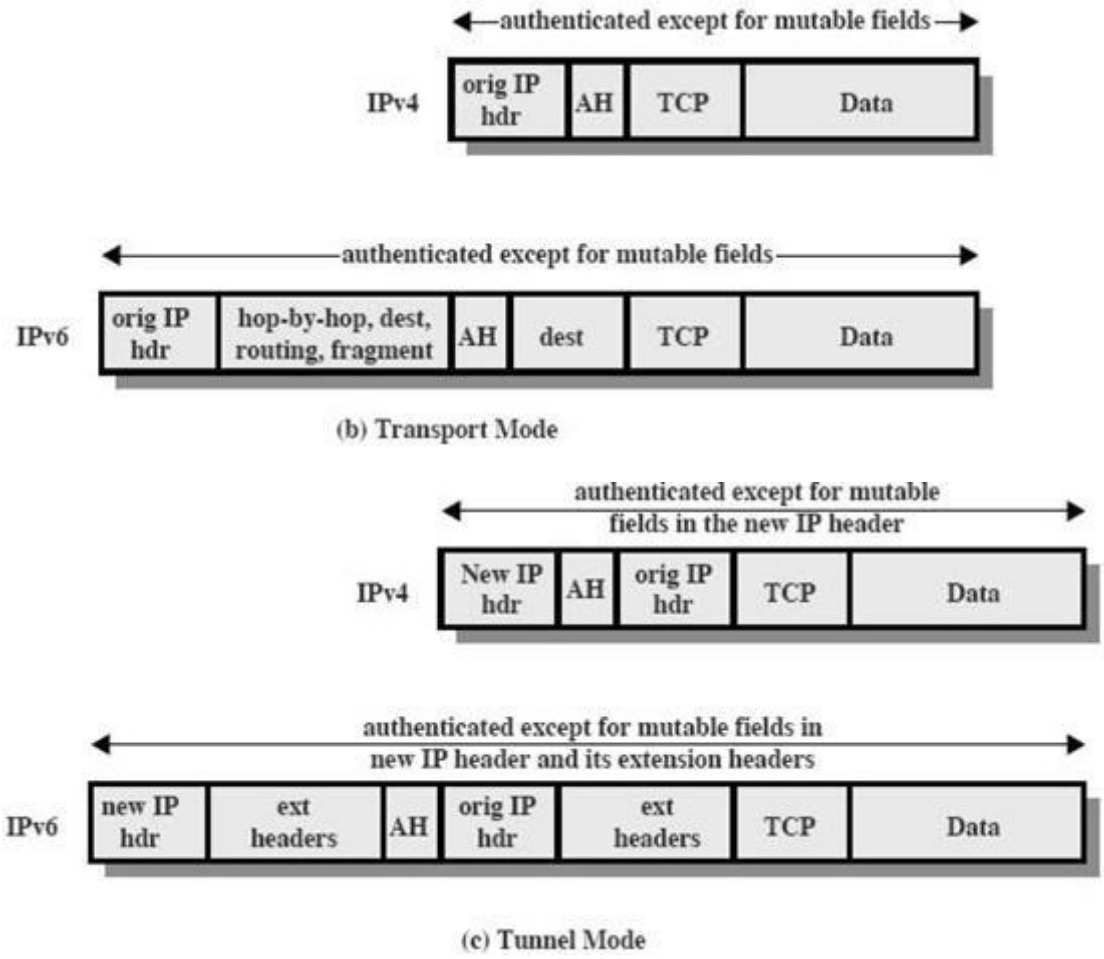
Transport and Tunnel Modes

The following figure shows typical IPv4 and IPv6 packets. In this case, the IP payload is a TCP segment; it could also be a data unit for any other protocol that uses IP, such as UDP or ICMP.



(a) Before Applying AH

For transport mode AH using IPv4, the AH is inserted after the original IP header and before the IP payload (e.g., a TCP segment) shown below. Authentication covers the entire packet, excluding mutable fields in the IPv4 header that are set to zero for MAC calculation. In the context of IPv6, AH is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the AH appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the AH header, depending on the semantics desired. Again, authentication covers the entire packet, excluding mutable fields that are set to zero for MAC calculation.



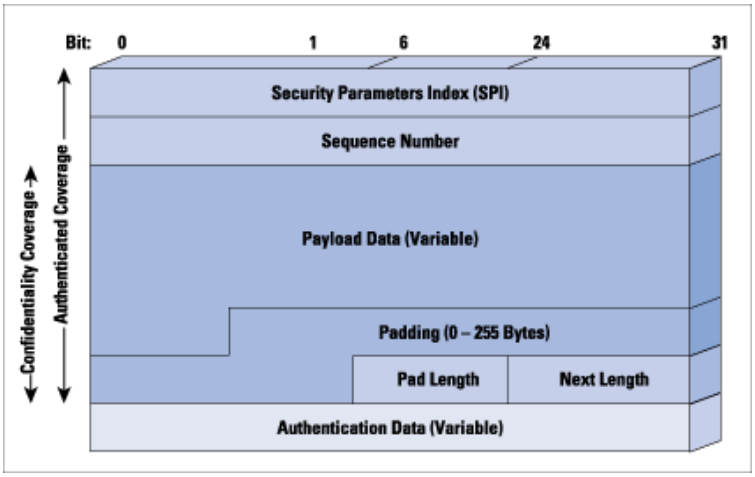
For tunnel mode AH, the entire original IP packet is authenticated, and the AH is inserted between the original IP header and a new outer IP header. The inner IP header carries the ultimate source and destination addresses, while an outer IP header may contain different IP addresses (e.g., addresses of firewalls or other security gateways). With tunnel mode, the entire inner IP packet, including the entire inner IP header is protected by AH. The outer IP header (and in the case of IPv6, the outer IP extension headers) is protected except for mutable and unpredictable fields.

ENCAPSULATING SECURITY PAYLOAD

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format

The following figure shows the format of an ESP packet. It contains the following fields:



Security Parameters Index (32 bits): Identifies a security association.

• **Sequence Number** (32 bits): A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.

• **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.

• **Padding (0-255 bytes):** This field is used to make the length of the plaintext to be a multiple of some desired number of bytes. It is also added to provide confidentiality.

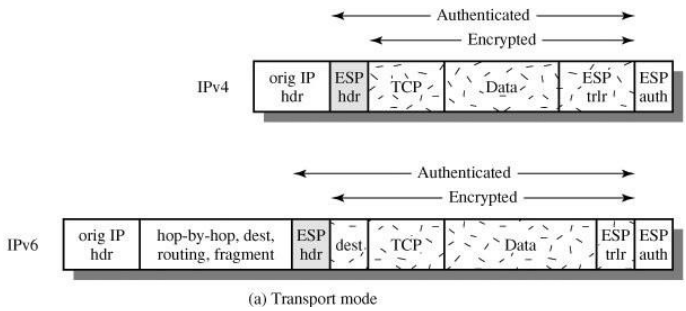
• **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.

• **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).

• **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

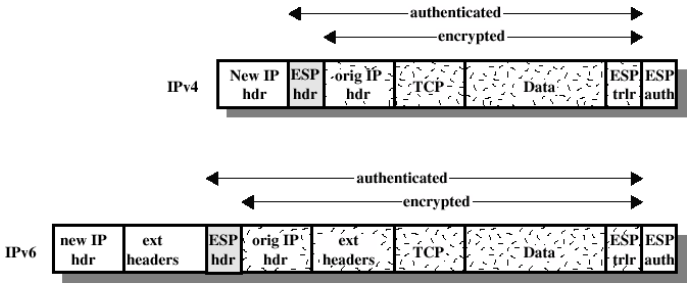
Adding encryption makes ESP a bit more complicated because the encapsulation *surrounds* the payload rather than *precedes* it as with AH: ESP includes header and trailer

Transport Mode ESP



(a) Transport mode

Tunnel Mode ESP

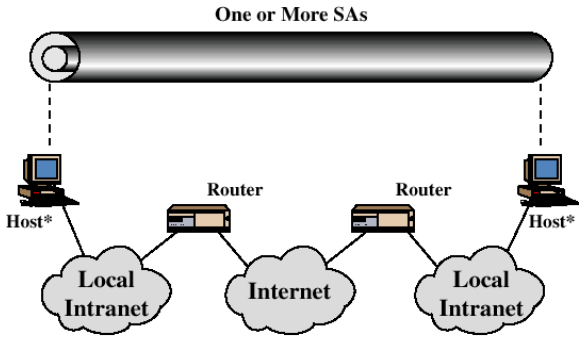


(b) Tunnel Mode

Basic Combinations of Security Associations

The IPSec Architecture document lists four examples of combinations of SAs that must be supported by compliant IPSec hosts (e.g., workstation, server) or security gateways (e.g. firewall, router).

case:-1



(a) Case 1

All security is provided between end systems that implement IPSec. For any two end systems to communicate via an SA, they must share the appropriate secret keys. Among the possible combinations:

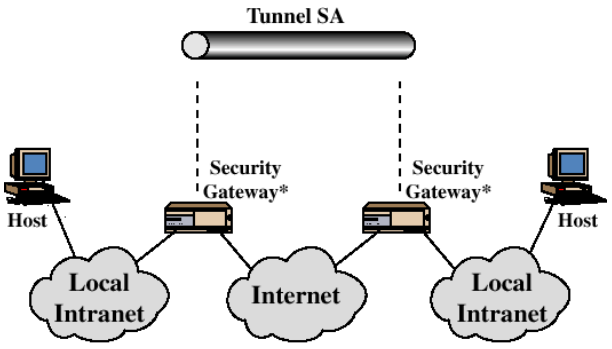
a) AH in transport mode

b) ESP in transport mode

c) ESP followed by AH in transport mode (an ESP SA inside an AH SA)

d) Any one of a, b, or c inside an AH or ESP in tunnel mode

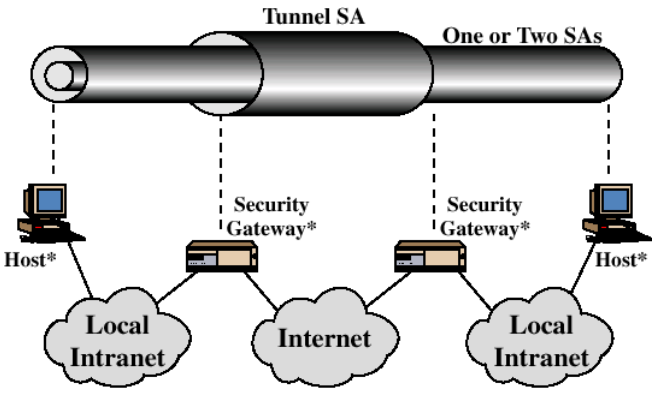
Case:-2



(b) Case 2

Security is provided only between gateways (routers, firewalls, etc.) and no hosts implement IPSec. This case illustrates simple virtual private network support. The security architecture document specifies that only a single tunnel SA is needed for this case. The tunnel could support AH, ESP, or ESP with the authentication option. Nested tunnels are not required because the IPSec services apply to the entire inner packet.

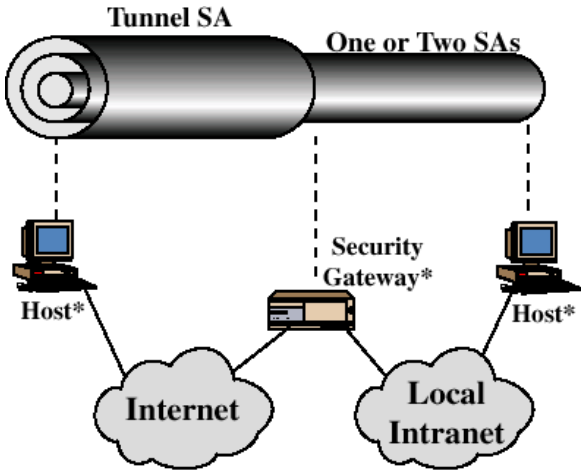
Case-3:-



(c) Case 3

The third combination is similar to the second, but in addition provides security even to nodes. This combination makes use of two tunnels first for gateway to gateway and second for node to node. Either authentication or the encryption or both can be provided by using gateway to gateway tunnel. An additional IPSec service is provided to the individual nodes by using node to node tunnel.

Case:-4



(d) Case 4

This combination is suitable for serving remote users i.e., the end user sitting anywhere in the world can use the internet to access the organizational workstations via the firewall. This combination states that only one tunnel is needed for communication between a remote user and an organizational firewall.

KEY MANAGEMENT

The key management portion of IPSec involves the determination and distribution of secret keys. The IPSec Architecture document mandates support for two types of key management:

• **Manual:** A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is practical for small, relatively static environments.

• **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system with an evolving configuration.

The default automated key management protocol for IPSec is referred to as ISAKMP/Oakley and consists of the following elements:

• **Oakley Key Determination Protocol**: Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate specific formats.

• **Internet Security Association and Key Management Protocol (ISAKMP):** ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.

Oakley Key Determination Protocol

Oakley is a refinement of the Diffie-Hellman key exchange algorithm. The Diffie-Hellmanalgorithm has two attractive features:

• Secret keys are created only when needed. There is no need to store secret keys for along period of time, exposing them to increased vulnerability.

• The exchange requires no pre-existing infrastructure other than an agreement on theglobal parameters.

However, Diffie-Hellman has got some weaknesses:

• No identity information about the parties is provided.

• It is possible for a man-in-the-middle attack

• It is computationally intensive. As a result, it is vulnerable to a clogging attack, in whichan opponent requests a high number of keys.

Oakley is designed to retain the advantages of Diffie-Hellman while countering its weaknesses.

Features of Oakley

The Oakley algorithm is characterized by five important features:

1. It employs a mechanism known as cookies to thwart clogging attacks.
2. It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.
3. It uses nonces to ensure against replay attacks.
4. It enables the exchange of Diffie-Hellman public key values.
5. It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

In clogging attacks, an opponent forges the source address of a legitimate user and sends a public Diffie-Hellman key to the victim. The victim then performs a modular exponentiation to compute the secret key. Repeated messages of this type can clog the victim's system with useless work. The **cookie exchange** requires that each side send a pseudorandom number, the cookie, in the initial message, which the other side acknowledges. This acknowledgment must be repeated in the first message of the Diffie-Hellman key exchange. The recommended method for creating the cookie is to perform a

fast hash (e.g., MD5) over the IP Source and Destination addresses, the UDP Source and Destination ports, and a locally generated secret value. Oakley supports the use of different **groups** for the Diffie-Hellman key exchange. Each group includes the definition of the two global parameters and the identity of the algorithm. Oakley employs **nonces** to ensure against replay attacks. Each nonce is a locally generated pseudorandom number. Nonces appear in responses and are encrypted during certain portions of the exchange to secure their use. Three different authentication methods can be used with Oakley are digital signatures, public-key encryption and Symmetric-key encryption.

Aggressive Oakley Key Exchange

Aggressive key exchange is a technique used for exchanging the message keys and is so called because only three messages are allowed to be exchanged at any time.

$\mathbf{I \rightarrow R:}\ CKY_I, OK\_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

$\mathbf{R \rightarrow I:}\ CKY_R, CKY_I, OK\_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

$\mathbf{I \rightarrow R:}\ CKY_I, CKY_R, OK\_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

Notation:

I	=	Initiator
R	=	Responder
CKY_I, CKY_R	=	Initiator, responder cookies
OK_KEYX	=	Key exchange message type
GRP	=	Name of Diffie-Hellman group for this exchange
g^x, g^y	=	Public key of initiator, responder; g^{xy} = session key from this exchange
EHAO, EHAS	=	Encryption, hash authentication functions, offered and selected
NIDP	=	Indicates encryption is not used for remainder of this message
ID_I, ID_R	=	Identifier for initiator, responder
N_I, N_R	=	Random nonce supplied by initiator, responder for this exchange
$S_{KI}[X], S_{KR}[X]$	=	Indicates the signature over X using the private key (signing key) of intiator, responder

Example of Aggressive Oakley Key Exchange

In the first step, the initiator (I) transmits a cookie, the group to be used, and I's public Diffie-Hellman key for this exchange. I also indicates the offered public-key encryption, hash, and authentication algorithms to be used in this exchange. Also included in this message are the identifiers of I and the responder (R) and I's nonce for this exchange. Finally, I appends a signature using I's private key that signs the two identifiers, the nonce, the group, the Diffie-Hellman public key, and the offered algorithms. When R receives the message, R verifies the signature using I's public signing key. R acknowledgesthe message by echoing back I's cookie, identifier, and nonce, as well as the group. R also includes in the message a cookie, R's Diffie-Hellman public key, the selected algorithms (which must be among the offered algorithms), R's identifier, and R's nonce for this exchange. Finally, R appends a signature using R's private key that signs the two

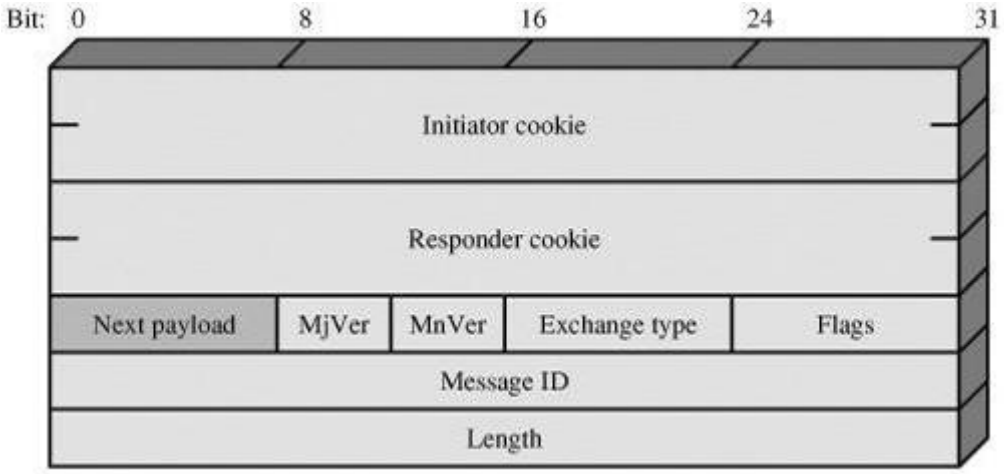
identifiers, the two nonces, the group, the two Diffie-Hellman public keys, and the selected algorithms.

When I receives the second message, I verifies the signature using R's public key. The nonce values in the message assure that this is not a replay of an old message. To complete the exchange, I must send a message back to R to verify that I has received R's public key.

ISAKMP

ISAKMP defines procedures and packet formats to establish, negotiate, modify, and delete security associations. As part of SA establishment, ISAKMP defines payloads for exchanging key generation and authentication data.

ISAKMP Header Format



(a) ISAKMP header

An ISAKMP message consists of an ISAKMP header followed by one or more payloads and must follow UDP transport layer protocol for its implementation. The header format of an ISAKMP header is shown below:

- Initiator Cookie (64 bits): Cookie of entity that initiated SA establishment, SA notification, or SA deletion.
- Responder Cookie (64 bits): Cookie of responding entity; null in first message from initiator.
- Next Payload (8 bits): Indicates the type of the first payload in the message
- Major Version (4 bits): Indicates major version of ISAKMP in use.
- Minor Version (4 bits): Indicates minor version in use.

• Exchange Type (8 bits): Indicates the type of exchange. Can be informational, aggressive, authentication only, identity protection or base exchange (S).

• Flags (8 bits): Indicates specific options set for this ISAKMP exchange. Two bits so far defined: The Encryption bit is set if all payloads following the header are encrypted using the encryption algorithm for this SA. The Commit bit is used to ensure that encrypted material is not received prior to completion of SA establishment.

• Message ID (32 bits): Unique ID for this message.

• Length (32 bits): Length of total message (header plus all payloads) in octets.

ISAKMP Payload Types

All ISAKMP payloads begin with the same generic payload header shown below.

(b) Generic payload header

The Next Payload field has a value of 0 if this is the last payload in the message; otherwise its value is the type of the next payload. The Payload Length field indicates the length in octets of this payload, including the generic payload header. There are many different ISAKMP payload types. They are:

a. The SA payload is used to begin the establishment of an SA. The Domain of Interpretation parameter identifies the DOI under which negotiation is taking place. The Situation parameter defines the security policy for this negotiation; in essence, the levels of security required for encryption and confidentiality are specified (e.g., sensitivity level, security compartment).

b. The Proposal payload contains information used during SA negotiation. The payload indicates the protocol for this SA (ESP or AH) for which services and mechanisms are being negotiated. The payload also includes the sending entity's SPI and the number of transforms. Each transform is contained in a transform payload.

c. The Transform payload defines a security transform to be used to secure the communications channel for the designated protocol. The Transform # parameter serves to identify this particular payload so that the responder may use it to indicate acceptance

of this transform. The Transform-ID and Attributes fields identify a specific transform (e.g., 3DES for ESP, HMAC-SHA-1-96 for AH) with its associated attributes (e.g., hash length).

d. The Key Exchange payload can be used for a variety of key exchange techniques, including Oakley, Diffie-Hellman, and the RSA-based key exchange used by PGP. The Key Exchange data field contains the data required to generate a session key and is dependenton the key exchange algorithm used.

e. The Identification payload is used to determine the identity of communicating peers and may be used for determining authenticity of information. Typically the ID Data field will contain an IPv4 or IPv6 address.

f. The Certificate payload transfers a public-key certificate. The Certificate Encoding field indicates the type of certificate or certificate-related information, which may include SPKI, ARL, CRL, PGP info etc. At any point in an ISAKMP exchange, the sender may include a Certificate Request payload to request the certificate of the other communicating entity.

g. The Hash payload contains data generated by a hash function over some part of the message and/or ISAKMP state. This payload may be used to verify the integrity of the data in a message or to authenticate negotiating entities.

h. The Signature payload contains data generated by a digital signature function over some part of the message and/or ISAKMP state. This payload is used to verify the integrityof the data in a message and may be used for nonrepudiation services.

i. The Nonce payload contains random data used to guarantee liveness during an exchange and protect against replay attacks.

j. The Notification payload contains either error or status information associated with this SA or this SA negotiation. Some of the ISAKMP error messages that have been definedare Invalid Flags, Invalid Cookie, Payload Malformed etc

k. The Delete payload indicates one or more SAs that the sender has deleted from its database and that therefore are no longer valid.

ISAKMP Exchanges

ISAKMP provides a framework for message exchange, with the payload types serving as the building blocks. The specification identifies five default exchange types that should be supported.

1. Base Exchange: allows key exchange and authentication material to be transmitted together. This minimizes the number of exchanges at the expense of not providing identity protection.

(a) Base Exchange

(1)**I**⟶**R**: SA; NONCE	Begin ISAKMP-SA negotiation
(2)**R**⟶**E**: SA; NONCE	Basic SA agreed upon
(3)**I**⟶**R**: KE; ID_I AUTH	Key generated; Initiator identity verified by responder
(4)**R**⟶**E**: KE; ID_R AUTH	Responder identity verified by initiator; Key generated; SA established

The first two messages provide cookies and establish an SA with agreed protocol and transforms; both sides use a nonce to ensure against replay attacks. The last two messages exchange the key material and user IDs, with an authentication mechanism used to authenticate keys, identities, and the nonces from the first two messages.

2. Identity Protection Exchange: expands the Base Exchange to protect the users' identities.

(b) Identity Protection Exchange

(1)**I**⟶**R**: SA	Begin ISAKMP-SA negotiation
(2)**R**⟶**E**: SA	Basic SA agreed upon
(3)**I**⟶**R**: KE; NONCE	Key generated
(4)**R**⟶**E**: KE; NONCE	Key generated
(5)*__I__⟶**R**: ID_I; AUTH	Initiator identity verified by responder
(6)*__R__⟶**E**: ID_R; AUTH	Responder identity verified by initiator; SA established

The first two messages establish the SA. The next two messages perform key exchange, with nonces for replay protection. Once the session key has been computed, the two parties

exchange encrypted messages that contain authentication information, such as digital signatures and optionally certificates validating the public keys.

3. Authentication Only Exchange: used to perform mutual authentication, without a key exchange

The first two messages establish the SA. In addition, the responder uses the second message to convey its ID and uses authentication to protect the message. The initiator sends the third message to transmit its authenticated ID.

4. Aggressive Exchange: minimizes the number of exchanges at the expense of notproviding identity protection.

(d) Aggressive Exchange

(1)**I**⟶**R**: SA; KE; NONCE; ID_I;	Begin ISAKMP-SA negotiation and key exchange
(2)**R**⟶**E**: SA; KE; NONCE; ID_R; AUTH	Initiator identity verified by responder; Key generated; Basic SA agreed upon
(3)*__I__⟶**R**: AUTH	Responder identity verified by initiator; SA established

In the first message, the initiator proposes an SA with associated offered protocol and transform options. The initiator also begins the key exchange and provides its ID. In the second message, the responder indicates its acceptance of the SA with a particular protocol and transform, completes the key exchange, and authenticates the transmitted information. In the third message, the initiator transmits an authentication result that covers the previous information, encrypted using the shared secret session key.

5. Informational Exchange: used for one-way transmittal of information for SA management.

(e) Informational Exchange

(1)*__I__⟶**R**: N/D	Error or status notification, or deletion

UNIT-5

Web Security: *Web Security Considerations, Secure Socket Layer (SSL) and Transport Layer Security (TLS), Secure Electronic Transaction (SET).* ***Firewalls:*** *Firewall Design Principles, Types of Firewalls.*

Usage of internet for transferring or retrieving the data has got many benefits like speed, reliability, security etc. Much of the Internet's success and popularity lies in the fact that it is an open global network. At the same time, the fact that it is open and globalmakes it not very secure. The unique nature of the Internet makes exchanging information and transacting business over it inherently dangerous. The faceless, voiceless, unknown entities and individuals that share the Internet may or may not be who or what they profess to be. In addition, because the Internet is a global network, it does not recognize national borders and legal jurisdictions. As a result, the transacting parties may not be where they say they are and may not be subject to the same laws or regulations.

For the exchange of information and for commerce to be secure on any network, especially the Internet, a system or process must be put in place that satisfies requirements for confidentiality, access control, authentication, integrity, and nonrepudiation. These requirements are achieved on the Web through the use of encryption and by employing digital signature technology. There are many examples on the Web of the practical application of encryption. One of the most important is the SSL protocol.

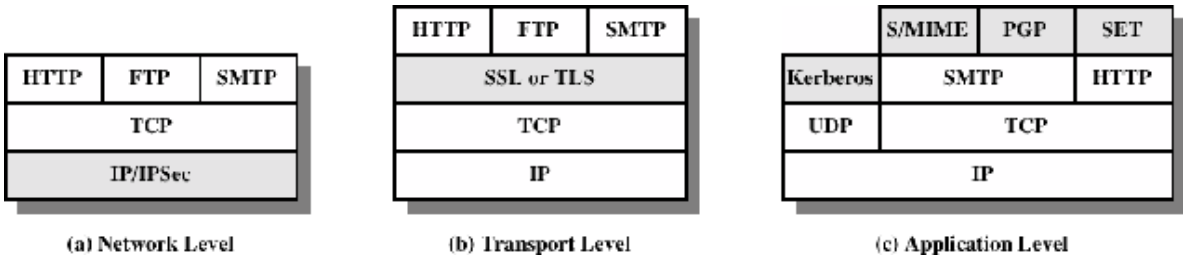
A summary of types of security threats faced in using the Web is given below:

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

One way of grouping the security threats is in terms of passive and active attacks. *Passive attacks* include eavesdropping on network traffic between browser and server and gaining access to information on a website that is supposed to be restricted. *Active attacks* include impersonating another user, altering messages in transit between client and server and altering information on a website. Another way of classifying these security threats is in terms of location of the threat: Web server, Web browser and network traffic between browser and server.

Web Traffic Security Approaches

Various approaches for providing Web Security are available, where they are similar in the services they provide and also similar to some extent in the mechanisms they use. They differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack. The main approaches are IPSec, SSL or TLS and SET.



Relative location of Security Faculties in the TCP/IP Protocol Stack

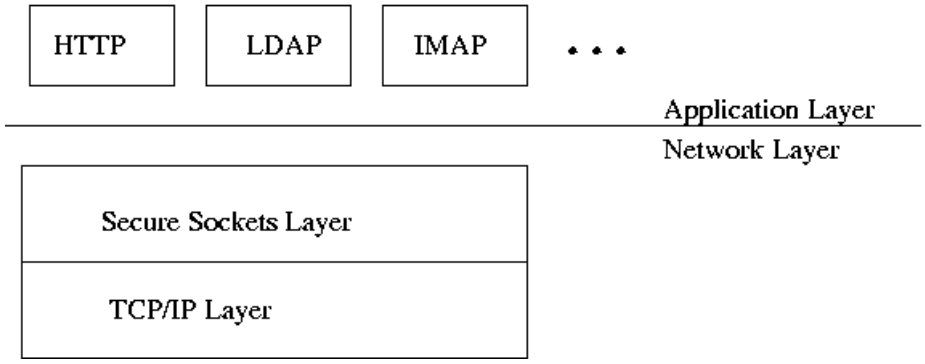
IPSec provides security at the network level and the main advantage is that it is transparent to end users and applications. In addition, IPSec includes a filtering capability so that only selected traffic can be processed. **Secure Socket Layer or Transport Layer Security (SSL/TLS)** provides security just above the TCP at transport layer. Two implementation choices are present here. Firstly, the SSL/TLS can be implemented as a

part of TCP/IP protocol suite, thereby being transparent to applications. Alternatively, SSL can be embedded in specific packages like SSL being implemented by Netscape and Microsoft Explorer browsers. **Secure Electronic Transaction (SET)** approach provides application-specific services i.e., according to the security requirements of a particular application. The main advantage of this approach is that service can be tailored to the specific needs of a given application.

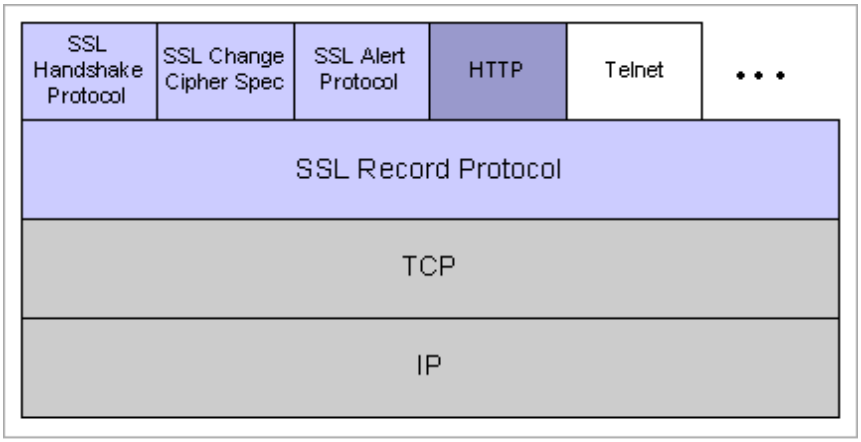
SECURE SOCKET LAYER/TRANSPORT LAYER SECURITY

SSL was developed by Netscape to provide security when transmitting information on the Internet. The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP).

SSL runs above TCP/IP and below high-level application protocols



SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity and encryption for privacy. SSL protocol has different versions such as SSLv2.0, SSLv3.0, where SSLv3.0 has an advantage with the addition of support for certificate chain loading. SSL 3.0 is the basis for the Transport Layer Security [TLS] protocol standard. SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol, but rather two layers of protocols as shown below:



SSL Protocol Stack

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

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

• **Session**: An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

An SSL session is *stateful*. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states. An SSL session may include multiple secure connections; in addition, parties may have multiple simultaneous sessions.

A session state is defined by the following parameters:

Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

Peer certificate: An X509.v3 certificate of the peer. This element of the state may be null.

Compression method: The algorithm used to compress data prior to encryption.

Cipher spec: Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.

Master secret: 48-byte secret shared between the client and server.

Is resumable: A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters:

Server and client random: Byte sequences that are chosen by the server and client for each connection.

Server write MAC secret: The secret key used in MAC operations on data sent by the server.

Client write MAC secret: The secret key used in MAC operations on data sent by the client.

Server write key: The conventional encryption key for data encrypted by the server and decrypted by the client.

Client write key: The conventional encryption key for data encrypted by the client and decrypted by the server.

***Initialization vectors*:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV with the following record.

Sequence numbers: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264-1.

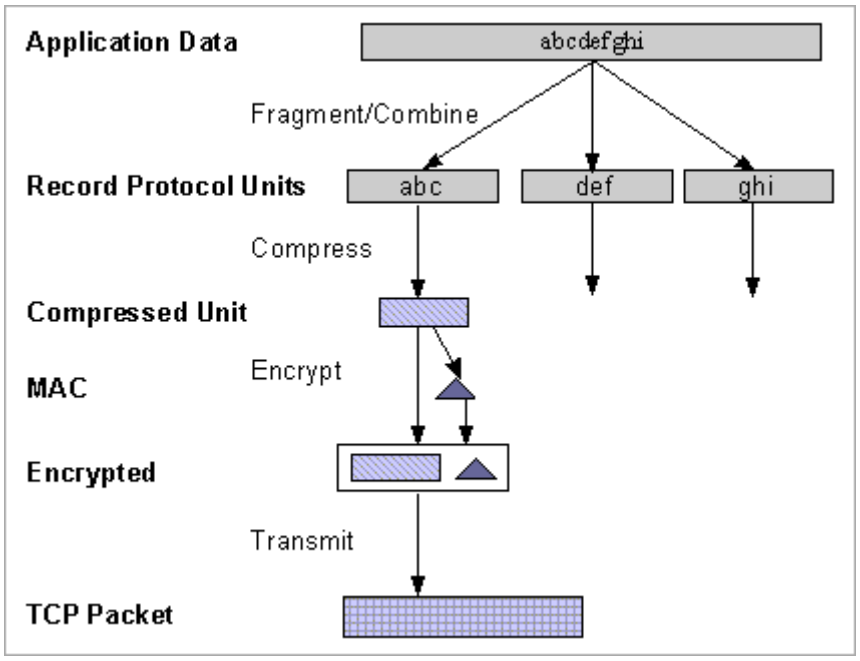
SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

• Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.

• Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

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



The first step is fragmentation. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. The next step in processing is to compute a message authentication code over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as:

hash(MAC_write_secret || pad_2 ||

hash(MAC_write_secret || pad_1 || seq_num ||

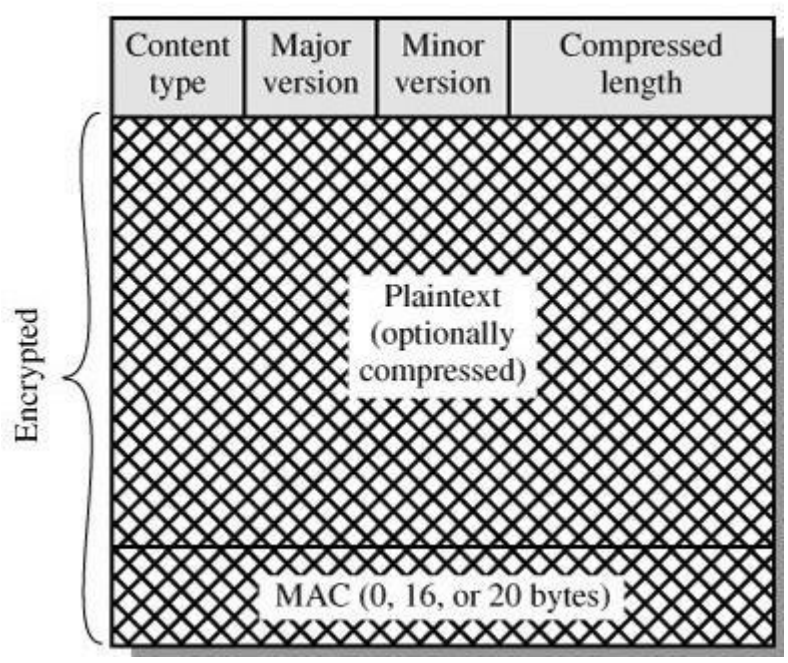
SSLCompressed.type ||

SSLCompressed.length || SSLCompressed.fragment)) Where,

MAC_write_secret = Secret shared key pad_1	=	the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times for
pad_2	=	the byte 0x5C (0101 1100) repeated 48 times for MD5 and 40 times for SHA-1

The main difference between HMAC and above calculation is that the two pads are concatenated in SSLv3 and are XORed in HMAC. Next, the compressed message plus the MAC are encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed 214 + 2048. The encryption algorithms allowed are AES-128/256, IDEA-128, DES-40, 3DES-168, RC2-40, Fortezza, RC4-40 and RC4-128. For stream encryption, the compressed message plus the MAC are encrypted whereas, for block encryption, padding may be added after the MAC prior to encryption.

SSL Record Format



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

• Content Type (8 bits): The higher layer protocol used to process the enclosed fragment.

• Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.

• Minor Version (8 bits): Indicates minor version in use. For SSLv3, the value is 0.

• Compressed Length (16 bits): The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is 214 + 2048.

The content types that have been defined are change_cipher_spec, alert, handshake, and application_data.

SSL Change Cipher Spec Protocol

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message, which consists of a single byte with the value 1.

The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

SSL Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes.

The first byte takes the value warning(1) or fatal(2) to convey the severity of the message.If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. The fatal alerts are listedbelow

• unexpected_message: An inappropriate message was received.

• bad_record_mac: An incorrect MAC was received.

• decompression_failure: The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).

• handshake_failure: Sender was unable to negotiate an acceptable set of security parameters given the options available.

• illegal_parameter: A field in a handshake message was out of range or inconsistent withother fields.

The remainder of the alerts are given below:

• close_notify: Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.

• no_certificate: May be sent in response to a certificate request if no appropriate certificate is available.

• bad_certificate: A received certificate was corrupt (e.g., contained a signature that did not verify).

• unsupported_certificate: The type of the received certificate is not supported.

• certificate_revoked: A certificate has been revoked by its signer.

• certificate_expired: A certificate has expired.

• certificate_unknown: Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

SSL Handshake Protocol

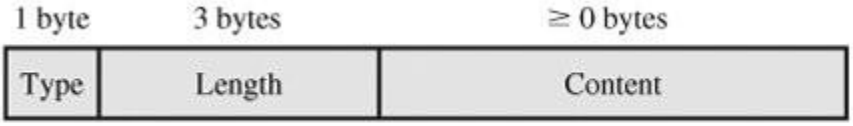
SSL Handshake protocol ensures establishment of reliable and secure session between client and server and also allows server & client to:

• authenticate each other

• to negotiate encryption & MAC algorithms

• to negotiate cryptographic keys to be used

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown below and each message has three fields:



(c) Handshake Protocol

• **Type (1 byte**): Indicates one of 10 messages.

• **Length (3 bytes):** The length of the message in bytes.

• **Content (>=0 bytes):** The parameters associated with this message

The following figure shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.in phases o Establish Security Capabilities

o Server Authentication and Key Exchange

o Client Authentication and Key Exchange

o Finish

Phase 1. Establish Security Capabilities

This phase is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message with the following parameters:

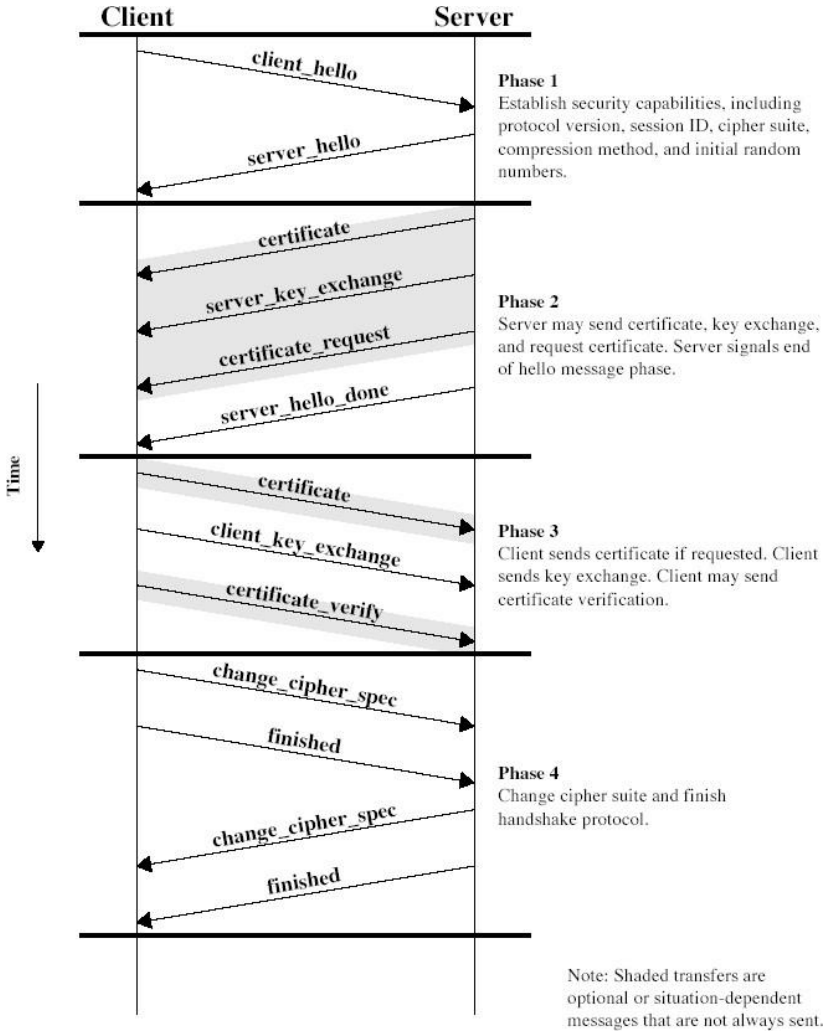
• Version: The highest SSL version understood by the client.

• Random: A client-generated random structure, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.

Session ID: A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.

• CipherSuite: This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec.

• Compression Method: This is a list of the compression methods the client supports.



Phase 2. Server Authentication and Key Exchange

The server begins this phase by sending its certificate via a certificate message, which contains one or a chain of X.509 certificates. The **certificate message** is required for any agreed-on key exchange method except anonymous Diffie-Hellman. Next, a **server_key_exchange** message may be sent if it is required. It is not required in two instances: (1) The server has sent a certificate with fixed Diffie-Hellman parameters, or (2) RSA key exchange is to be used.

Phase 3. Client Authentication and Key Exchange

Once the server_done message is received by client, it should verify whether a valid certificate is provided and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. If the server has requested a certificate, the client begins this phase by sending a **certificate message**. If no suitable certificate is available, the client sends a no_certificate alert instead. Next is the **client_key_exchange** message, for which the content of the message depends on the type of key exchange.

Phase 4. Finish

This phase completes the setting up of a secure connection. The client sends a **change_cipher_spec** message and copies the pending CipherSpec into the current CipherSpec. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

TRANSPORT LAYER SECURITY

TLS was released in response to the Internet community's demands for a standardized protocol. TLS (Transport Layer Security), defined in RFC 2246, is a protocol for establishing a secure connection between a client and a server. TLS (Transport Layer Security) is capable of authenticating both the client and the server and creating a encrypted connection between the two. Many protocols use TLS (Transport Layer Security) to establish secure connections, including HTTP, IMAP, POP3, and SMTP. The TLS Handshake Protocol first negotiates key exchange using an asymmetric algorithm such as RSA or Diffie-Hellman. The TLS Record Protocol then begins opens an encrypted channel using a symmetric algorithm such as RC4, IDEA, DES, or 3DES. The TLS Record Protocol is also responsible for ensuring that the communications are not altered in transit. Hashing algorithms such as MD5 and SHA are used for this purpose. RFC 2246 is very similar to SSLv3. There are some minor differences ranging from protocol version numbers to generation of key material.

Version Number: The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 1.

Message Authentication Code: Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. For TLS, the MAC calculation encompasses the fields indicated in the following expression:

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

The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field TLSCompressed.version, which is the version of the protocol being employed.

Pseudorandom Function: TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The PRF is based on the following data expansion function:

P_hash(secret, seed) = HMAC_hash(secret, A(1) ||
seed) ||HMAC_hash(secret, A(2) || seed) ||
HMAC_hash(secret, A(3) || seed) || ...

where A() is defined as

A(0) = seed
A(i) = HMAC_hash (secret, A(i - 1))

The data expansion function makes use of the HMAC algorithm, with either MD5 or SHA-1 as the underlying hash function. As can be seen, P_hash can be iterated as many times as necessary to produce the required quantity of data. each iteration involves two executions of HMAC, each of which in turn involves two executions of the underlying hashalgorithm.

SET (SECURE ELECTRONIC TRANSACTION)

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. SET is not itself a payment system. Rather it is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network, such as the Internet, in a secure fashion. In essence, SET provides three services:

• Provides a secure communications channel among all parties involved in a transaction

• Provides trust by the use of X.509v3 digital certificates

• Ensures privacy because the information is only available to parties in a transactionwhen and where necessary

SET Requirements

- Provide confidentiality of payment and ordering information
- Ensure the integrity of all transmitted data
- Provide authentication that a cardholder is a legitimate user of a credit card account
- Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution
- Ensure the use of the best security practices and system design techniques to protectall legitimate parties in an electronic commerce transaction
- Create a protocol that neither depends on transport security mechanisms norprevents their use
- Facilitate and encourage interoperability among software and network providers

SET Key Features

To meet the requirements, SET incorporates the following features:

• Confidentiality of information

• Integrity of data

• Cardholder account authentication

• Merchant authentication

SET Participants

- Cardholder: purchasers interact with merchants from personal computers over theInternet
- Merchant: a person or organization that has goods or services to sell to the cardholder
- Issuer: a financial institution, such as a bank, that provides the cardholder with the payment card.
- Acquirer: a financial institution that establishes an account with a merchant andprocesses payment card authorizations and payments
- Payment gateway: a function operated by the acquirer or a designated third party thatprocesses merchant payment messages
- Certification authority (CA): an entity that is trusted to issue X.509v3 public-keycertificates for cardholders, merchants, and payment gateways

Events in a transaction

1. The customer obtains a credit card account with a bank that supports electronic payment and SET

2. The customer receives a X.509v3 digital certificate signed by the bank.

3. Merchants have their own certificates

4. The customer places an order

5. The merchant sends a copy of its certificate so that the customer can verify that it's avalid store

6. The order and payment are sent

7. The merchant requests payment authorization

8. The merchant confirms the order

9. The merchant ships the goods or provides the service to the customer

10. The merchant requests payment

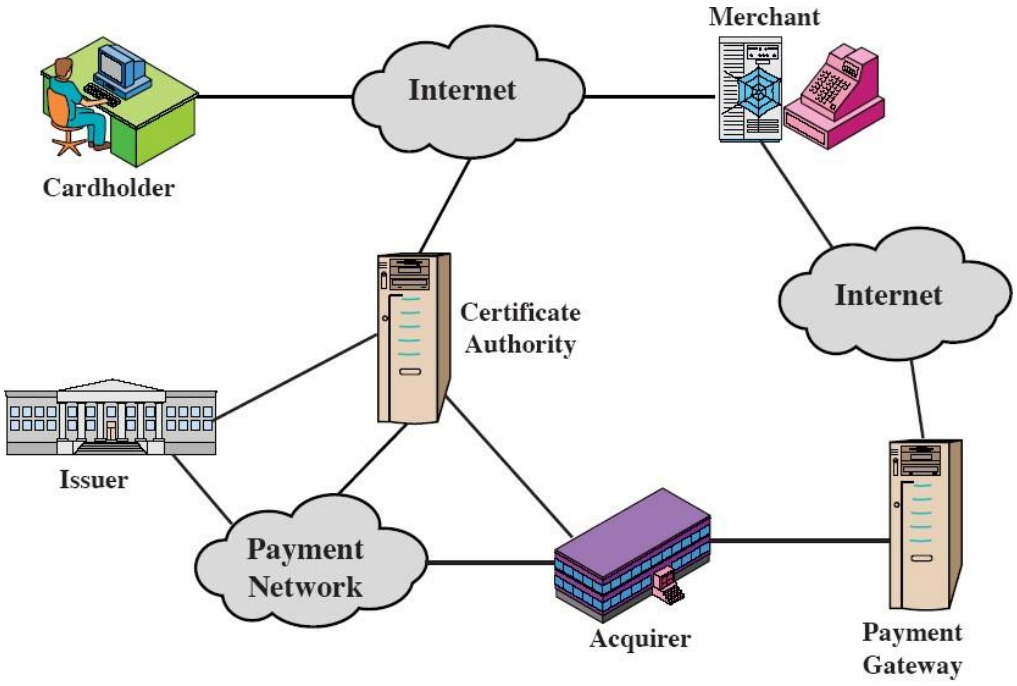
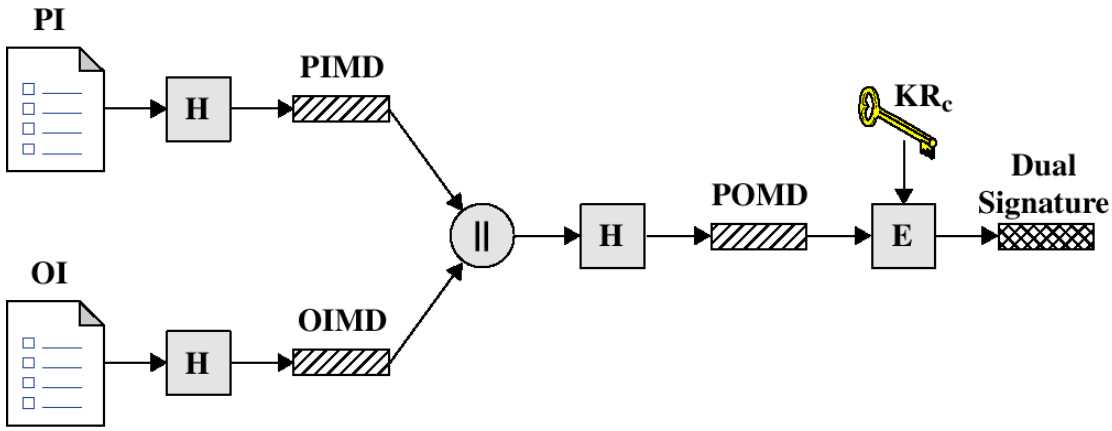


Figure 17.8 Secure Electronic Commerce Components

DUAL SIGNATURE

The purpose of the dual signature is to link two messages that are intended for two different recipients. The customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to

know the customer's credit card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate. The two items must be linked and the link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service.



PI = Payment Information
OI = Order Information
H = Hash function (SHA-1)
|| = Concatenation
PIMD = PI message digest
OIMD = OI message digest
POMD = Payment Order message digest
E = Encryption (RSA)
KR_c = Customer's private signature key

The customer takes the hash (using SHA-1) of the PI and the hash of the OI. These two hashes are then concatenated and the hash of the result is taken. Finally, the customer encrypts the final hash with his or her private signature key, creating the dual signature.
The operation can be summarized as

$$DS = E_{KR_c}[H(H(PI) \| \mathrm{H}(\mathrm{OI}))]$$

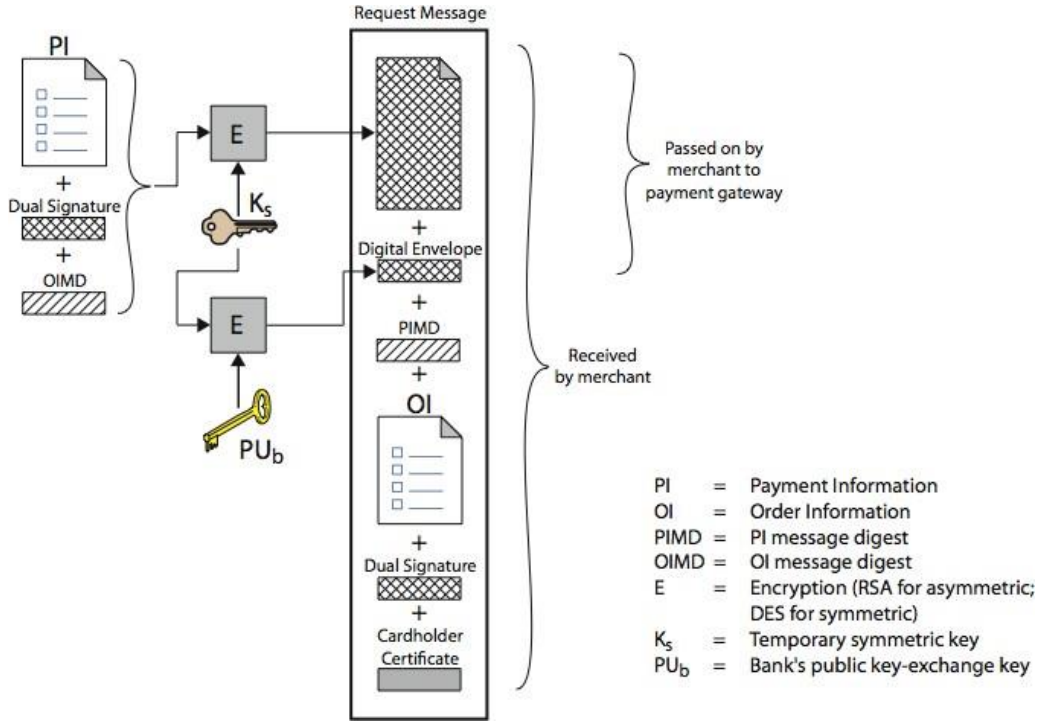
where KRc is the customer's private signature key. Now suppose that the merchant is in possession of the dual signature (DS), the OI, and the message digest for the PI (PIMD). The merchant also has the public key of the customer, taken from the customer's certificate. Then the merchant can compute the quantities H(PIMS||H[OI]) and DKUc(DS) where KUc is the customer's public signature key. If these two quantities are equal, then the merchant has verified the signature. Similarly, if the bank is in possession of DS, PI, the message digest for OI (OIMD), and the customer's public key, then the bank can
compute H(H[OI]||OIMD) and DKUc(DS). Again, if these two quantities are equal, then the

bank has verified the signature. To summarize:

- The merchant has received OI and verified the signature.
- The bank has received PI and verified the signature.
- The customer has linked the OI and PI and can prove the linkage.

For a merchant to substitute another OI, he has to find another OI whose hash exactly matches OIMD, which is deemed impossible. So, the OI cannot be linked with another PI.

Purchase Request

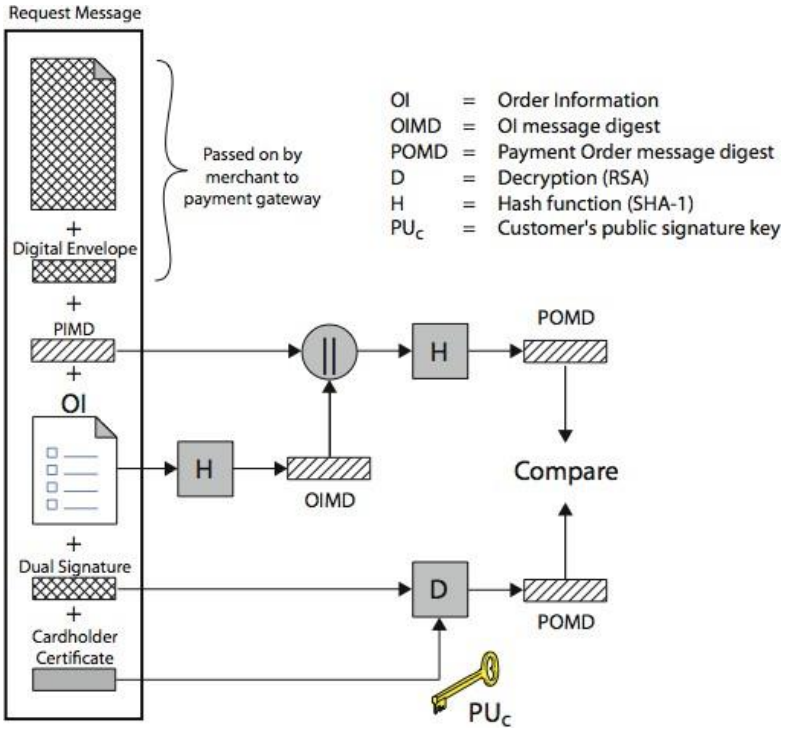


The message includes the following:

1. Purchase-related information, which will be forwarded to the payment gateway by the merchant and consists of: PI, dual signature & OI message digest (OIMD). These are encrypted using Ks. A digital envelope is also present which is formed by encrypting Ks with the payment gateway's public key-exchange key.
2. Order-related information, needed by the merchant and consists of: OI, dual signature, PI message digest (PIMD). OI is sent in the clear.
3. Cardholder certificate. This contains the cardholder's public signature key. It is needed by the merchant and payment gateway.

Merchant receives the Purchase Request message, the following actions are done:

1. verifies cardholder certificates using CA sigs
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization
4. sends a purchase response to cardholder



The Purchase Response message includes a response block that acknowledges the order and references the corresponding transaction number. This block is signed by the merchant using its private signature key. The block and its signature are sent to the customer, along with the merchant’s signature certificate. Necessary action will be takenby cardholder’s software upon verification of the certificates and signature.

FIREWALLS

A firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter, forming a single choke point where security and audit can be imposed. A firewall:

1. Defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.

2. provides a location for monitoring security-related events

3. is a convenient platform for several Internet functions that are not security related, such as NAT and Internet usage audits or logs

4. A firewall can serve as the platform for IPSec to implement virtual private networks.

Design Goals of Firewalls

All traffic from inside to outside must pass through the firewall (physically blocking all access to the local network except via the firewall)

Only authorized traffic (defined by the local security police) will be allowed to pass

The firewall itself is immune to penetration (use of trusted system with a secure operating system)

The four general techniques that firewalls use to control access and enforce the sites security policies are:

- Service control: Determines the types of Internet services that can be accessed, inbound or outbound
- Direction control: Determines the direction in which particular service requests are allowed to flow
- User control: Controls access to a service according to which user is attempting to access it
- Behavior control: Controls how particular services are used (e.g. filter e-mail)

The limitations of Firewalls are:

1. Cannot protect against attacks that bypass the firewall, eg PCs with dial-out capability to an ISP, or dial-in modem pool use.

2. do not protect against internal threats, eg disgruntled employee or one who cooperates with an attacker

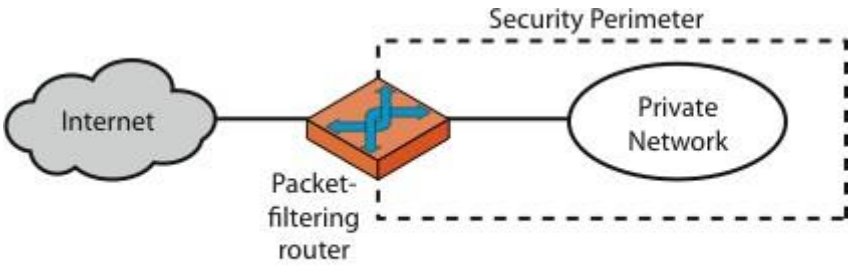
3. cannot protect against the transfer of virus-infected programs or files, given wide variety of O/S & applications supported

Types of Firewalls

Firewalls are generally classified as three types: packet filters, application-level gateways, & circuit-level gateways.

Packet-filtering Router

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet to forward or discard the packet. Filtering rules are based on information contained in a network packet such as src & dest IP addresses, ports, transport protocol & interface.



(a) Packet-filtering router

If there is no match to any rule, then one of two default policies are applied:

- that which is not expressly permitted is prohibited (default action is discard packet),conservative policy
- that which is not expressly prohibited is permitted (default action is forward packet), permissive policy

The default discard policy is more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are more likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security; the security administrator must, in essence, react to each new security threat as it becomes known. One advantage of a

packet-filtering router is its simplicity. Also, packet filters typically are transparent to users and are very fast.

The table gives some examples of packet-filtering rule sets. In each set, the rules are applied top to bottom.

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

A. Inbound mail is allowed to a gateway host only (port 25 is for SMTP incoming

B. explicit statement of the default policy

C. tries to specify that any inside host can send mail to the outside, but has problem that an outside machine could be configured to have some other application linked to port 25

D. properly implements mail sending rule, by checking ACK flag of a TCP segment is set

E. this rule set is one approach to handling FTP connections

Some of the attacks that can be made on packet-filtering routers & countermeasures are:

IP address spoofing: where intruder transmits packets from the outside with internal host source IP addresses, need to filter & discard such packets

Source routing attacks: where source specifies the route that a packet should take to bypass security measures, should discard all source routed packets

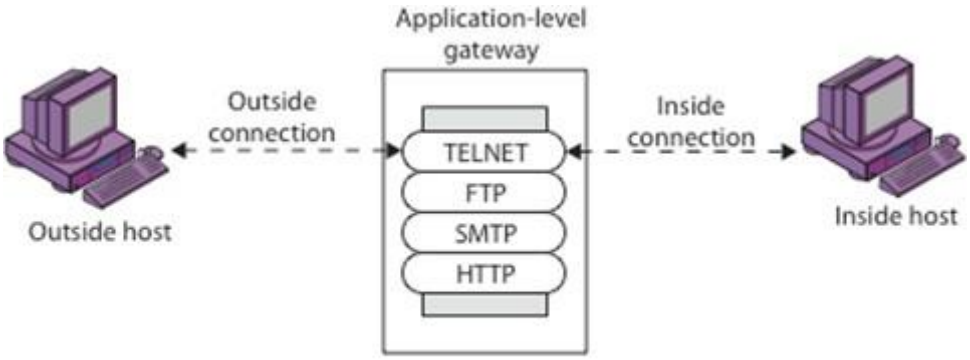
Tiny fragment attacks: intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into separate fragments to circumvent filtering rules needing full header info, can enforce minimum fragment size to include full header.

Stateful Packet Filters

A traditional packet filter makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context. A stateful inspection packet filter tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, and will allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory. Hence they are better able to detect bogus packets sent out of context.

APPLICATION LEVEL GATEWAY

An application-level gateway (or proxy server), acts as a relay of application-level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.



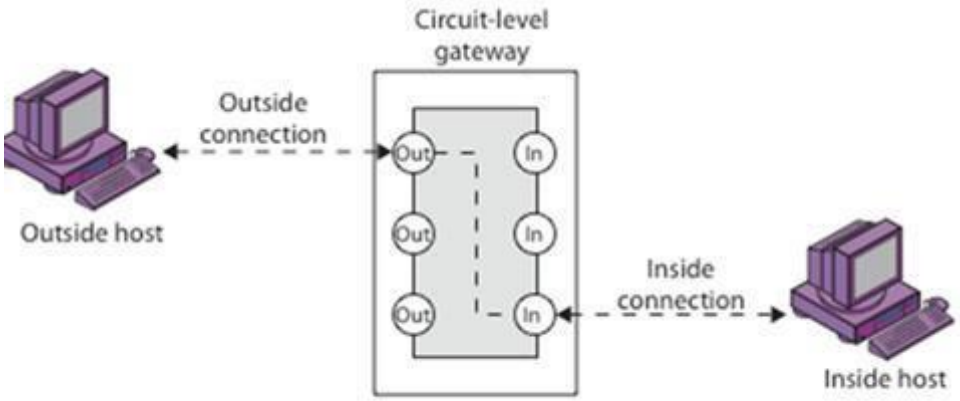
(b) Application-level gateway

Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level. A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

CIRCUIT LEVEL GATEWAY

A circuit-level gateway relays two TCP connections, one between itself and an inside TCP user, and the other between itself and a TCP user on an outside host. Once the two connections are established, it relays TCP data from one connection to the other without examining its contents. The security function consists of determining which connections will be allowed. It is typically used when internal users are trusted to decidewhat external services to access.

One of the most common circuit-level gateways is SOCKS, defined in RFC 1928. It consists of a SOCKS server on the firewall, and a SOCKS library & SOCKS-aware applications on internal clients. The protocol described here is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall. The protocol is conceptually a "shim-layer" between the application layer and the transport layer, and as such does not provide network-layer gateway services, such as forwarding of ICMP messages.



(c) Circuit-level gateway

Bastion Host

A bastion host is a critical strong point in the network's security, serving as a platform for an application-level or circuit-level gateway, or for external services. It is thus potentially exposed to "hostile" elements and must be secured to withstand this. Commoncharacteristics of a bastion host include that it:

- executes a secure version of its O/S, making it a trusted system
- has only essential services installed on the bastion host

- may require additional authentication before a user is allowed access to the proxy services
- is configured to support only a subset of the standard application's command set, with access only to specific hosts
- maintains detailed audit information by logging all traffic
- has each proxy module a very small software package specifically designed for network security
- has each proxy independent of other proxies on the bastion host
- have a proxy performs no disk access other than to read its initial configuration file
- have each proxy run as a non-privileged user in a private and secured directory
- A bastion host may have two or more network interfaces (or ports), and must be trusted to enforce trusted separation between these network connections, relaying traffic only according to policy.

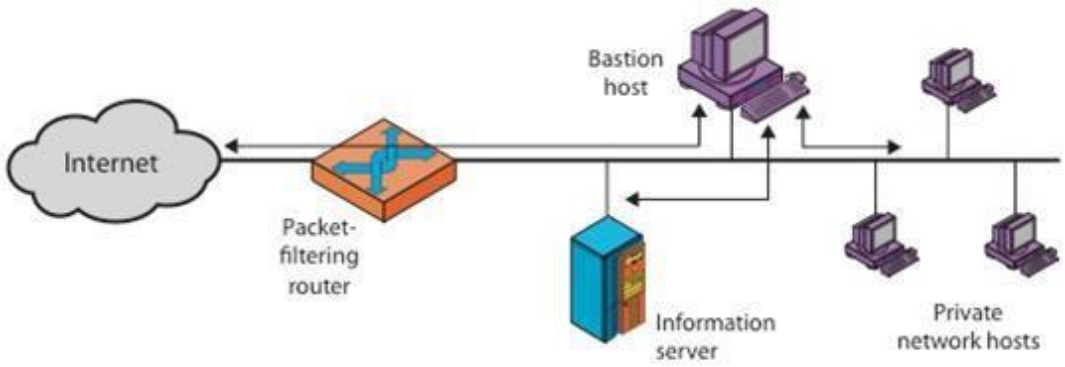
Firewall Configurations

In addition to the use of a simple configuration consisting of a single system, more complex configurations are possible and indeed more common. There are three common firewall configurations.

The following figure shows the "**screened host firewall, single-homed bastion configuration**", where the firewall consists of two systems:

• a packet-filtering router - allows Internet packets to/from bastion only

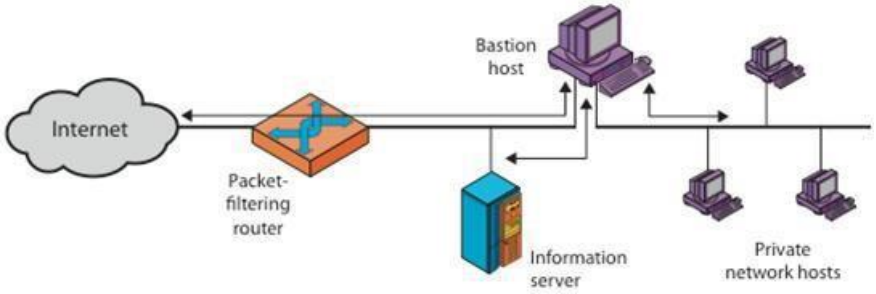
• a bastion host - performs authentication and proxy functions



(a) Screened host firewall system (single-homed bastion host)

This configuration has greater security, as it implements both packet-level & application-level filtering, forces an intruder to generally penetrate two separate systems to compromise internal security, & also affords flexibility in providing direct Internet access to specific internal servers (eg web) if desired.

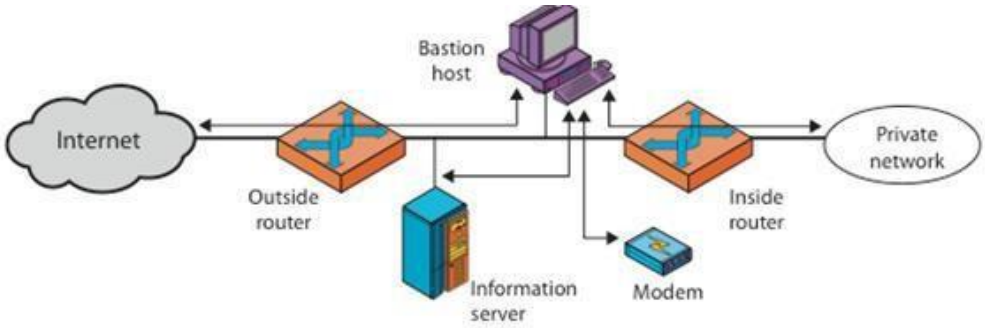
The next configuration illustrates the "**screened host firewall, dual-homed bastion configuration**" which physically separates the external and internal networks, ensuring two systems must be compromised to breach security. The advantages of dual layers of security are also present here.



(b) Screened host firewall system (dual-homed bastion host)

Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy, but are now separated from the internal network.

The third configurations illustrated below shows the "**screened subnet firewall configuration**", being the most secure shown.



(c) Screened-subnet firewall system

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- network. This may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked.

This configuration offers several advantages:

• There are now three levels of defense to thwart intruders

• The outside router advertises only the existence of the screened subnet to the Internet; therefore the internal network is invisible to the Internet

• Similarly, the inside router advertises only the existence of the screened subnet to the internal network; hence systems on the inside network cannot construct direct routes to the Internet