

ANNAMALAI  **UNIVERSITY**
(Accredited with 'A' Grade by NAAC)

FACULTY OF SCIENCE

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

M.Sc. (COMPUTER SCIENCE)

II YEAR – IV SEMESTER

PCSE402- NETWORK SECURITY

Elective - II

Network Security

Unit-I - Introduction - Attacks, Services, and Mechanisms – Security Attacks: Passive Attacks –Active Attacks – Security Services – Model for Internetwork Security – Internet Standards and RFCs: Internet Society – RFC Publication – Standardization Process – Non- Standardization Track Documents – Conventional Encryption and Message Confidentiality: Conventional Encryption Principles: Cryptography – Cryptanalysis – Feistel Cipher Structure – Conventional Encryption Algorithms – Cipher Blocks Models of Operation: Cipher Block Chaining Mode – Cipher Feedback Model – Location of Encryption Devices – Key distribution.

Unit-II - Public Key Cryptography and Message Authentication - Approaches to Message Authentication: Authentication using Conventional Encryption – Message Authentication without Message Encryption – Message Authentication Code – One Way Hash Function – Secure Hash Function and HMAC: Hash Function Requirements – Simple Hash Functions the SHA-1 Secure Hash Function – Other Secure Hash Functions – HMAC Design Objectives – HMAC Algorithm – Public Key Cryptography Principles: Public Key Encryption Structure Application for Public Key Cryptosystems – Requirements for Public Key Cryptography – Public Key Cryptography Algorithms: RSA Public Key Encryption Algorithms – Diffie-Hellman Key Exchange – Other Public Key Cryptography Algorithms – Digital Signature – Key Management.

Unit-III - Electronic Mail Security - Pretty Good Privacy (PGP): Notation – Operational Description – Cryptography Keys and Key Rings – Public Key Management – S/MIME: RFC 822 – Multipurpose Internet Mail Extensions – S/MIME Functionality – S/MIME Messages – S/MIME Certificate Processing – Enhanced Security Services – IP Security: IP Security Overview: Application of IPsec – Benefits of IPsec – Routing Applications – IP Security Architecture: IPsec Documents – IPsec Services – Security Associations – Transport and Tunnel Modes – Authentication Header – Encapsulating Security Payload – Combining Security Associations – Key Management: Oakley Key Determination Protocol – ISAKMP

Unit-IV - Web security - Web security Requirements: Web security Threats – Web Traffic Security Approaches – Secure Socket Layer (SSL) and Transport Layer Security (TLS): SSL Architecture – SSL Record Protocol – Change Cipher Spec Protocol – Alert Protocol - Handshake Protocol – Cryptographic Computations – Transport Layer Security – Secure Electronics Transaction (SET): SET Overview – Dual Signature – Payment Processing – Network Management Security: Basic Concepts of SNMP: Network Management Architecture – Network Management Protocol Architecture – Proxies – snmpv2 – snmpv1 Community Facility – snmpv3: SNMP Architecture – Message Processing and the User Security Model – View Based Access Control.

Unit-V - System Security - Intruders and Viruses: Intruders: Intrusion Techniques – Password Protection – Password Selection Strategies – Intrusions Detection – Viruses and Related Threats: Malicious Programs – The Nature of Viruses – Types of Viruses – Macro Viruses – Antiviruses Approaches – Advanced Antiviruses Techniques – Firewalls: Firewall Design Principles: Firewall Characteristics – Types of Firewalls – Firewall Configuration – Trusted Systems: Data Access Control – The Concept of Trusted Systems – Trojan Horse Defense.

Text book:

1. William Stallings, Network Security Essentials, Prentice Hall, Third Edition, 2006.

Reference Books:

1. Manish Tiwari, Handbook of Network Security and Anti Terrorism Laws, Neha Publishers and Distributors, 2012.
2. Adesh K. Pandey, Network Security and Administration, S.K. Kataria and Sons, 2010.

UNIT –I: INTRODUCTION

Network Security is used to protect both equipment and information. Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit.

Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

- **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers.
- **Network Security** - measures to protect data during their transmission.
- **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks.

SECURITY ATTACKS

There are four general categories of attack which are listed below.

Interruption

An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability e.g., destruction of piece of hardware, cutting of a communication line or Disabling of file management system.

Interception

An unauthorized party gains access to an asset. This is an attack on confidentiality. Unauthorized party could be a person, a program or a computer. e.g., wire tapping to capture data in the network, illicit copying of files.

Modification

An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. e.g., changing values in data file, altering a program, modifying the contents of messages being transmitted in a network.

Fabrication

An unauthorized party inserts counterfeit objects into the system. This is an attack on

authenticity. e.g., insertion of spurious message in a network or addition of records to a file.

CRYPTOGRAPHIC ATTACKS

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.

Passive attacks are of two types:

- **Release of message contents:** A telephone conversation, an e-mail message and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.
- **Traffic analysis:** If we had encryption protection in place, an opponent might still be able to observe the pattern of the message. The opponent could determine the location and identity of communication hosts and could observe the frequency and length of messages being exchanged.

Passive attacks are very difficult to detect because they do not involve any alteration of data. However, it is feasible to prevent the success of these attacks.

Active Attacks

These attacks involve some modification of the data stream or the creation of a false stream. These attacks can be classified in to four categories:

- **Masquerade** – One entity pretends to be a different entity.
- **Replay** – involves passive capture of a data unit and its subsequent transmission to produce an unauthorized effect.
- **Modification of messages** – Some portion of message is altered or the messages are delayed or recorded, to produce an unauthorized effect.
- **Denial of service** – Prevents or inhibits the normal use or management of communication
- **Facilities-** Another form of service denial is the disruption of an entire network, either by disabling the network or overloading it with messages so as to degrade performance.

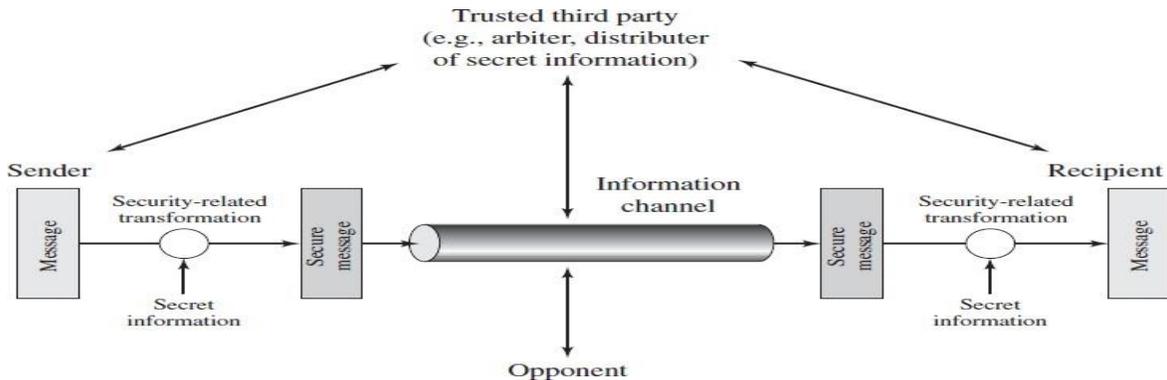
It is quite difficult to prevent active attacks absolutely, because to do so would require

physical protection of all communication facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them.

SECURITY SERVICES

- The classification of security services are as follows:
- **Confidentiality:** Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.
E.g. Printing, displaying and other forms of disclosure.
- **Authentication:** Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.
- **Integrity:** Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.
- **Non repudiation:** Requires that neither the sender nor the receiver of a message be able to deny the transmission.
- **Access control:** Requires that access to information resources may be controlled by or the target system.
- **Availability:** Requires that computer system assets be available to authorized parties when needed.

MODEL FOR NETWORK SECURITY



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

Symmetric cipher model consists of five components

1. Plain text: An Original message is known as plaintext. That is feed into the algorithm as input.
2. Encryption Algorithm: The encryption algorithm perform various transposition and substitution techniques.
3. Secret Key: The secret key is also input to the encryption algorithm. The value is independent of the plain text and encryption algorithm.
4. Cipher text: This is the scrambled message produced as output. It depends on the plaintext and secret key.
5. Decryption Algorithm: The encryption algorithm run in the reverse. It take the cipher text and the secret key and produce the original message.

The symmetric encryption is also known as Conventional encryption or single key encryption.

CRYPTOGRAPHY

Cryptographic systems are generally classified along three independent dimensions:

- **Type of operations used for transforming plain text to cipher text**

All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.

- **The number of keys used**

If the sender and receiver uses same key then it is said to be **symmetric key (or) single key (or) conventional encryption**.

If the sender and receiver use different keys then it is said to be **public key encryption**.

- **The way in which the plain text is processed**

A **block cipher** processes the input and block of elements at a time, producing output block for each input block.

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

Types of Cryptosystems

Fundamentally, there are two types of cryptosystems based on the manner in which encryption-decryption is carried out in the system –

- Symmetric Key Encryption
- Asymmetric Key Encryption

The main difference between these cryptosystems is the relationship between the encryption and the decryption key. Logically, in any cryptosystem, both the keys are closely associated. It is practically impossible to decrypt the ciphertext with the key that is unrelated to the encryption key.

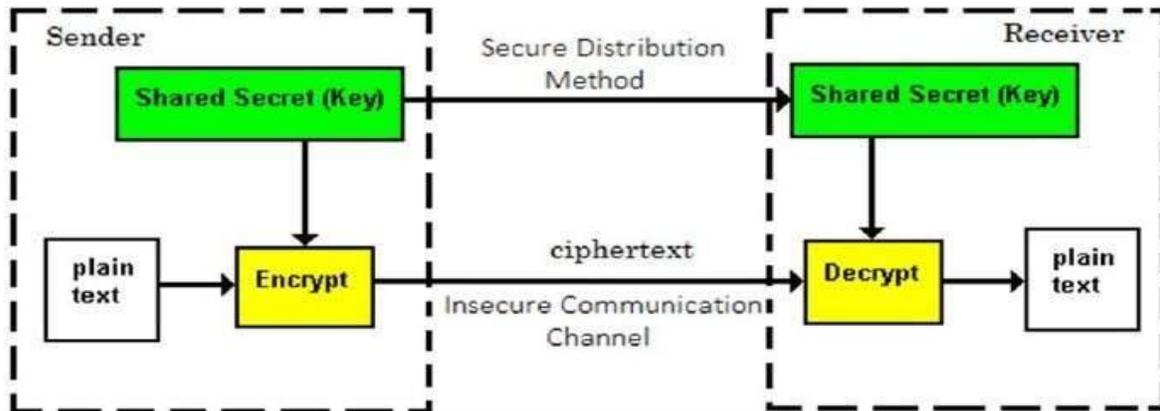
Symmetric Key Encryption

The encryption process where **same keys are used for encrypting and decrypting** the information is known as Symmetric Key Encryption.

The study of symmetric cryptosystems is referred to as **symmetric cryptography**.

Symmetric cryptosystems are also sometimes referred to as **secret key cryptosystems**.

A few well-known examples of symmetric key encryption methods are – Digital Encryption Standard (DES), Triple-DES (3DES), IDEA, and BLOWFISH.



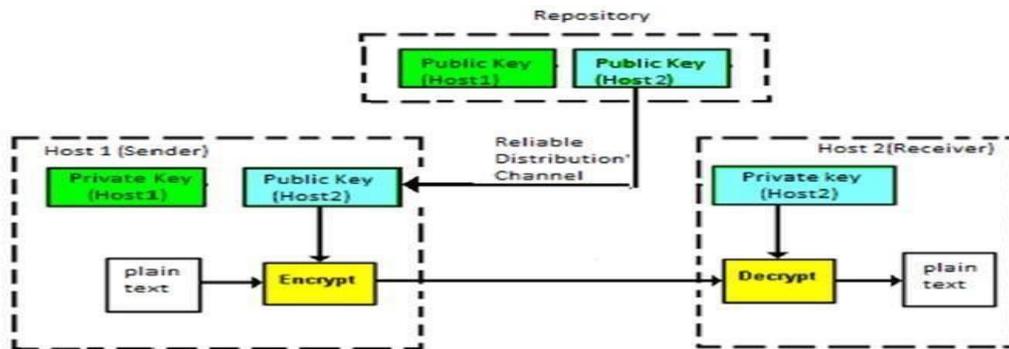
Prior to 1970, all cryptosystems employed symmetric key encryption. Even today, its relevance is very high and it is being used extensively in many cryptosystems. It is very unlikely that this encryption will fade away, as it has certain advantages over asymmetric key encryption.

The salient features of cryptosystem based on symmetric key encryption are –

- Persons using symmetric key encryption must share a common key prior to exchange of information.
- Keys are recommended to be changed regularly to prevent any attack on the system.
- A robust mechanism needs to exist to exchange the key between the communicating parties. As keys are required to be changed regularly, this mechanism becomes expensive and cumbersome.
- In a group of n people, to enable two-party communication between any two persons, the number of keys required for group is $n \times (n - 1)/2$.
- Length of Key (number of bits) in this encryption is smaller and hence, process of encryption-decryption is faster than asymmetric key encryption.
- Processing power of computer system required to run symmetric algorithm is less.

Asymmetric Key Encryption

The encryption process where **different keys are used for encrypting and decrypting the information** is known as Asymmetric Key Encryption. Though the keys are different, they are mathematically related and hence, retrieving the plaintext by decrypting ciphertext is feasible. The process is depicted in the following illustration –



Asymmetric Key Encryption was invented in the 20th century to come over the necessity of pre-shared secret key between communicating persons. The salient features of this encryption scheme are as follows –

- Every user in this system needs to have a pair of dissimilar keys, **private key** and **public key**. These keys are mathematically related – when one key is used for encryption, the other can decrypt the ciphertext back to the original plaintext.
- It requires to put the public key in public repository and the private key as a well-guarded secret. Hence, this scheme of encryption is also called **Public Key Encryption**.
- Though public and private keys of the user are related, it is computationally not feasible to find one from another. This is a strength of this scheme.
- When *Host1* needs to send data to *Host2*, he obtains the public key of *Host2* from repository, encrypts the data, and transmits.
- *Host2* uses his private key to extract the plaintext.
- Length of Keys (number of bits) in this encryption is large and hence, the process of encryption-decryption is slower than symmetric key encryption.
- Processing power of computer system required to run asymmetric algorithm is higher.

Cryptanalysis

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

- **Ciphertext Only Attacks (COA)** – In this method, the attacker has access to a set of ciphertext(s). He does not have access to corresponding plaintext. COA is said to be successful when the corresponding plaintext can be determined from a given set of ciphertext. Occasionally, the encryption key can be determined from this attack. Modern cryptosystems are guarded against ciphertext-only attacks.
- **Known Plaintext Attack (KPA)** – In this method, the attacker knows the plaintext for some parts of the ciphertext. The task is to decrypt the rest of the ciphertext using this information. This may be done by determining the key or via some other method. The best example of this attack is *linear cryptanalysis* against block ciphers.
- **Chosen Plaintext Attack (CPA)** – In this method, the attacker has the text of his choice encrypted. So he has the ciphertext-plaintext pair of his choice. This simplifies his task of determining the encryption key. An example of this attack is *differential cryptanalysis* applied against block ciphers as well as hash functions. A popular public key cryptosystem, RSA is also vulnerable to chosen-plaintext attacks.
- **Dictionary Attack** – This attack has many variants, all of which involve compiling a ‘dictionary’. In simplest method of this attack, attacker builds a dictionary of ciphertexts and corresponding plaintexts that he has learnt over a period of time. In future, when an attacker gets the ciphertext, he refers the dictionary to find the corresponding plaintext.
- **Brute Force Attack (BFA)** – In this method, the attacker tries to determine the key by attempting all possible keys. If the key is 8 bits long, then the number of possible keys is $2^8 = 256$. The attacker knows the ciphertext and the algorithm, now he attempts all the 256 keys one by one for decryption. The time to complete the attack would be very high if the key is long.

- **Birthday Attack** – This attack is a variant of brute-force technique. It is used against the cryptographic hash function. When students in a class are asked about their birthdays, the answer is one of the possible 365 dates. Let us assume the first student's birthdate is 3rd Aug. Then to find the next student whose birthdate is 3rd Aug, we need to enquire $1.25^* \cdot \sqrt{365} \approx 25$ students.

Similarly, if the hash function produces 64 bit hash values, the possible hash values are 1.8×10^{19} . By repeatedly evaluating the function for different inputs, the same output is expected to be obtained after about 5.1×10^9 random inputs.

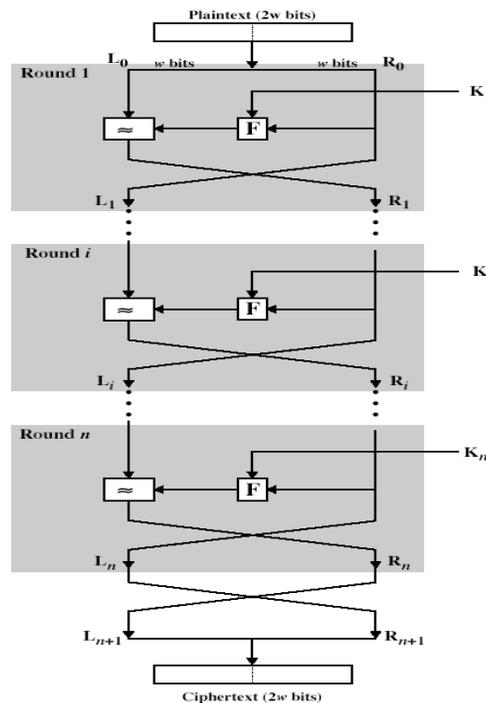
If the attacker is able to find two different inputs that give the same hash value, it is a **collision** and that hash function is said to be broken.

- **Man in Middle Attack (MIM)** – The targets of this attack are mostly public key cryptosystems where key exchange is involved before communication takes place.
 - Host *A* wants to communicate to host *B*, hence requests public key of *B*.
 - An attacker intercepts this request and sends his public key instead.
 - Thus, whatever host *A* sends to host *B*, the attacker is able to read.
 - In order to maintain communication, the attacker re-encrypts the data after reading with his public key and sends to *B*.
 - The attacker sends his public key as *A*'s public key so that *B* takes it as if it is taking it from *A*.
- **Side Channel Attack (SCA)** – This type of attack is not against any particular type of cryptosystem or algorithm. Instead, it is launched to exploit the weakness in physical implementation of the cryptosystem.
- **Timing Attacks** – They exploit the fact that different computations take different times to compute on processor. By measuring such timings, it is possible to know about a particular computation the processor is carrying out. For example, if the encryption takes a longer time, it indicates that the secret key is long.
- **Power Analysis Attacks** – These attacks are similar to timing attacks except that the amount of power consumption is used to obtain information about the nature of the underlying computations.

- **Fault analysis Attacks** – In these attacks, errors are induced in the cryptosystem and the attacker studies the resulting output for useful information.

FEISTEL CIPHER STRUCTURE

The input to the encryption algorithm are a plaintext block of length $2w$ bits and a key K . The plaintext block is divided into two halves L_0 and R_0 . The two halves of the data pass through „ n “ rounds of processing and then combine to produce the ciphertext block. Each round „ i “ has inputs L_{i-1} and R_{i-1} , derived from the previous round, as well as the subkey K_i , derived from the overall key K . In general, the subkeys K_i are different from K and from each other.



Classical Feistel Network

All rounds have the same structure. A substitution is performed on the left half of the data (as similar to S-DES). This is done by applying a round function F to the right half of the data and then taking the XOR 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 . Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network. The exact realization of a Feistel network depends on the choice of the following parameters and design features:

Block size - Increasing size improves security, but slows cipher

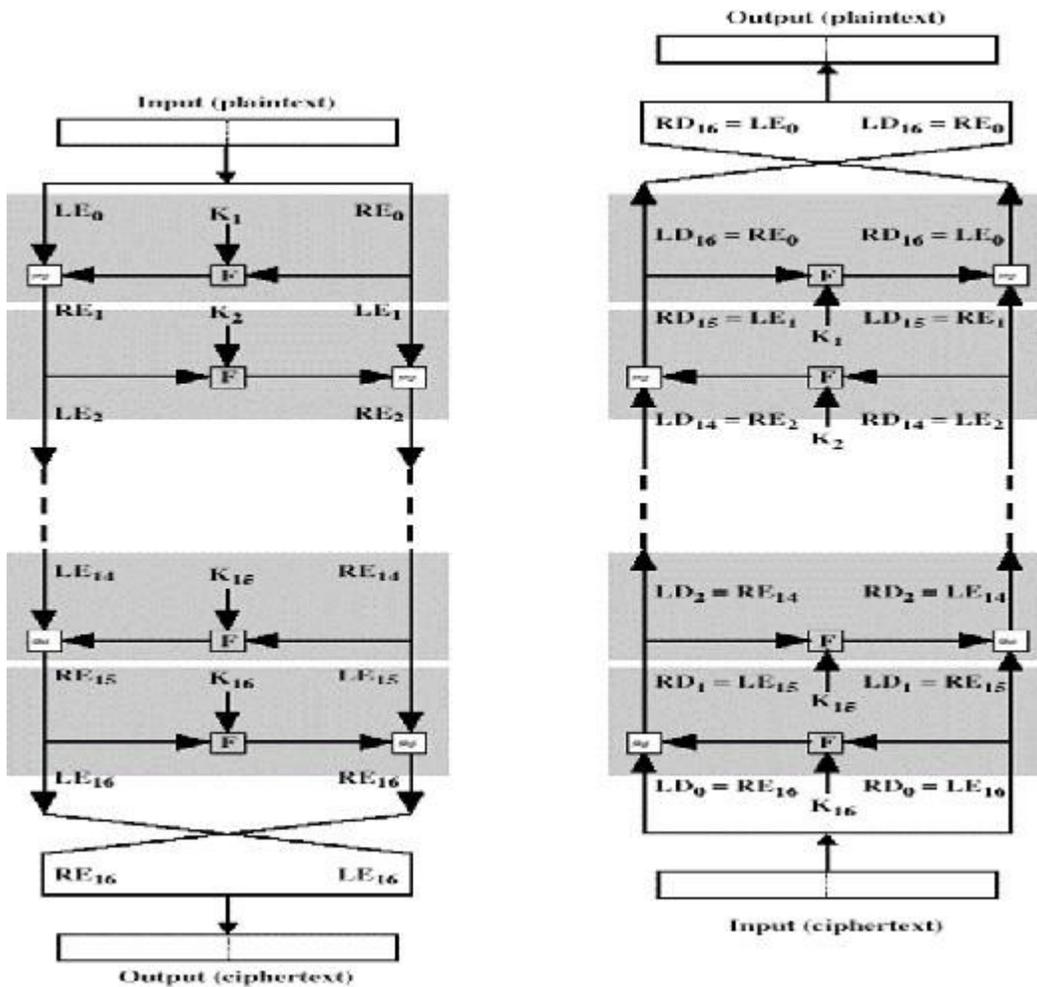
Key size - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher

Number of rounds - Increasing number improves security, but slows cipher

Subkey generation - Greater complexity can make analysis harder, but slows cipher

Round function - Greater complexity can make analysis harder, but slows cipher

Fast software en/decryption & ease of analysis - are more recent concerns for practical use and testing.



Feistel encryption and decryption

The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey k_i in reverse order. i.e., k_n in the first round, k_{n-1} in second round and so on. For clarity, we use the notation LE_i and RE_i for data traveling through the decryption algorithm. The diagram below indicates that,

at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the value swapped.

i.e., $RE_i \parallel LE_i$ (or) equivalently $RD_{16-i} \parallel LD_{16-i}$

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is $RE_{16} \parallel LE_{16}$. The output of that round is the cipher text. Now take the cipher text and use it as input to the same algorithm. The input to the first round is $RE_{16} \parallel LE_{16}$, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.

CLASSICAL ENCRYPTION TECHNIQUES

There are two basic building blocks of all encryption techniques: substitution and transposition.

SUBSTITUTION TECHNIQUES

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 (or) shift 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 3 places further down the alphabet.

e.g., plain text : pay more money

Cipher text: SDB PRUH PRQHB

Note that the alphabet is wrapped around, so that letter following „z“ is „a“. For each plaintext letter p , substitute the cipher text letter c such that

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

A shift may be any amount, so that general Caesar algorithm is

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

Where k takes on a value in the range 1 to 25. The decryption algorithm is simply $P =$

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

Monoalphabetic Ciphers

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Recall the assignment for the Caesar cipher:

plain: a b c d e f g h i j k l m n o p q r s t u v w x y z

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

If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are $26!$ or greater than 4×10^{26} possible keys. This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis. Such an approach is referred to as a **monoalphabetic substitution cipher**.

Playfair cipher

The best known multiple letter encryption cipher is the playfair, which treats diagrams in the plaintext as single units and translates these units into cipher text diagrams. The playfair algorithm is based on the use of 5×5 matrix of letters constructed using a keyword. Let the keyword be „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 alphabetical order. The letter „i“ and „j“ count as one letter. Plaintext is encrypted two letters at a time.

According to the following rules:

- Repeating plaintext letters that would fall in the same pair are separated with a Filler letter such as „x“.
- 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 following the last.
- Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last.

Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

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

Plaintext : meet me at the school house

Splitting two letters as a unit => me et me at th es ch o x ol ho us ex
 => CL KL CL RS PD IL HY AV MP HF XL IU

Polyalphabetic ciphers

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

- A set of related monoalphabetic substitution rules are used
- A key determines which particular rule is chosen for a given transformation.

Vigenere cipher

In this scheme, the set of related monoalphabetic substitution rules consisting of 26 caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter.

e.g., Caesar cipher with a shift of 3 is denoted by the key value 'd' (since a=0, b=1, c=2 and so on). To aid in understanding the scheme, a matrix known as vigenere tableau is constructed. Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top.

The process of encryption is simple: Given a key letter X and a plaintext letter y, the cipher text is at the intersection of the row labeled x and the column labeled y; in this case, the ciphertext is V. To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword.

e.g., key = d e c e p t i v e d e c e p t i v e d e c e p t i v e
 P T = w e a r e d i s c o v e
 r e d s a v e y o u r s e l f
 C T = Z I C V T W Q N G R Z G V T W A V Z H C Q Y G L M G J

Decryption is equally simple. The key letter again identifies the row. The position of the cipher text letter in that row determines the column, and the plaintext letter is at the top of that column.

Strength of Vigenere cipher

- o There are multiple cipher text letters for each plaintext letter.
- o Letter frequency information is obscured.

TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

Rail fence is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Plaintext = meet at the school house

To encipher this message with a rail fence of depth 2, we write the message as follows:

m e a t e c o l o s e t t h s h o h u e

The encrypted message is MEATECOLOSETTHSHOHUE

Row Transposition Ciphers-

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 columns then becomes the key of the algorithm.

e.g., plaintext = meet at the school house Key = 4 3 1 2 5 6 7

PT = m e e t a t t h e s c h o o l h o u s e

CT = ESOTCUEEHMHLAHSTOETO

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed.

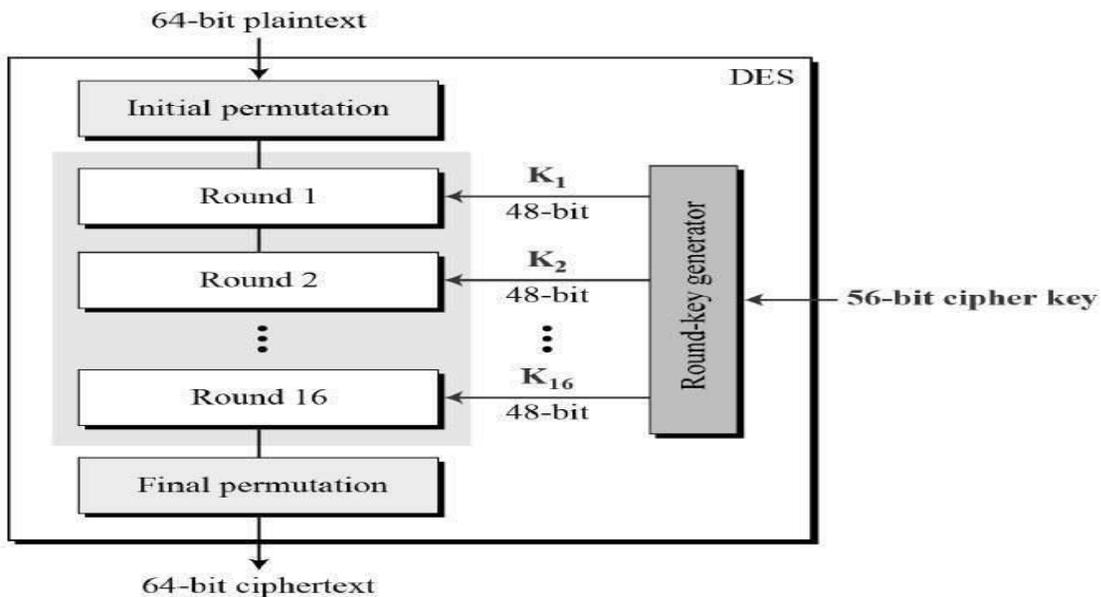
DATA ENCRYPTION STANDARD (DES)

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

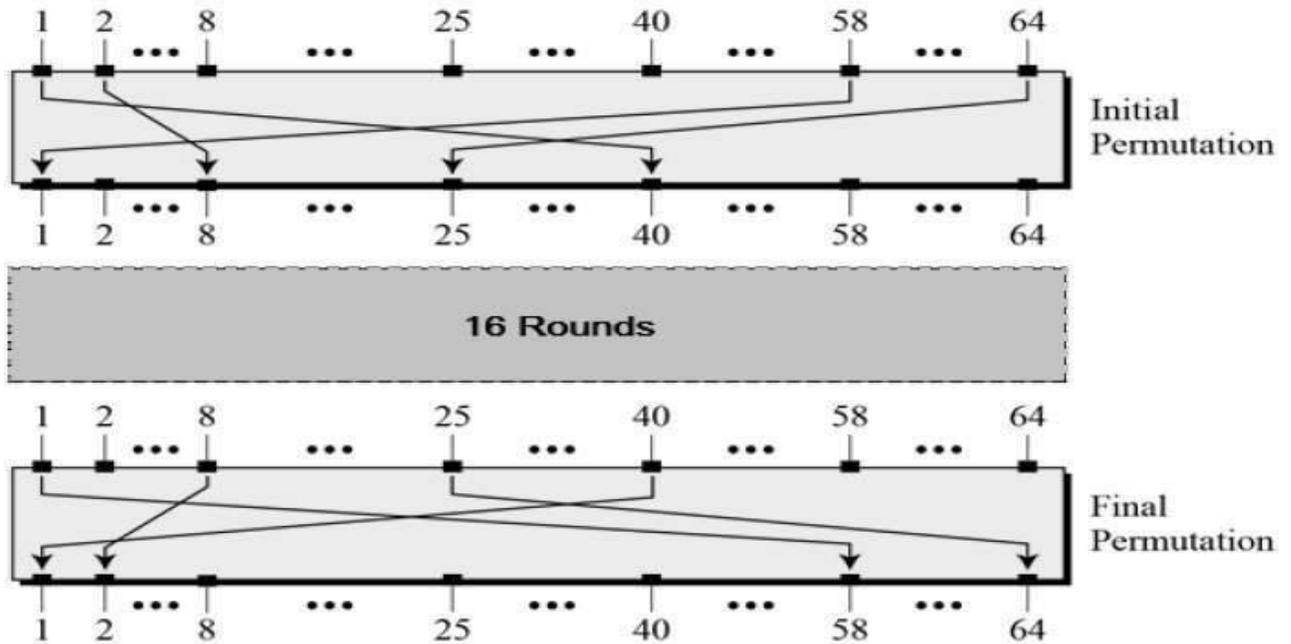
Since DES is based on the Feistel Cipher, all that is required to specify DES is –

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



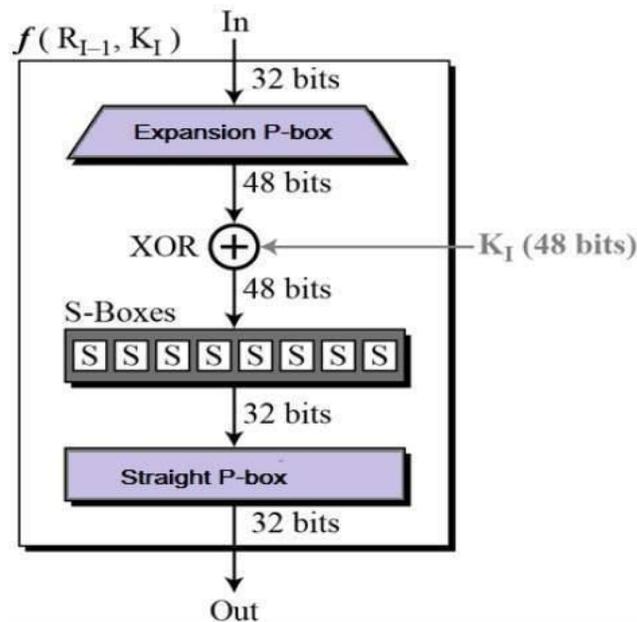
Initial and Final Permutation

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 –

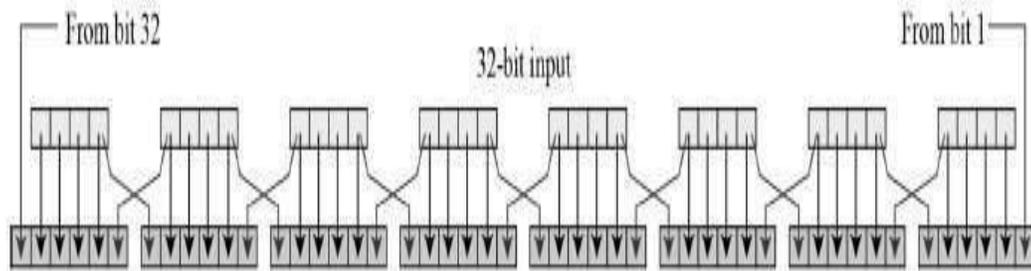


Round Function

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



- **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 –

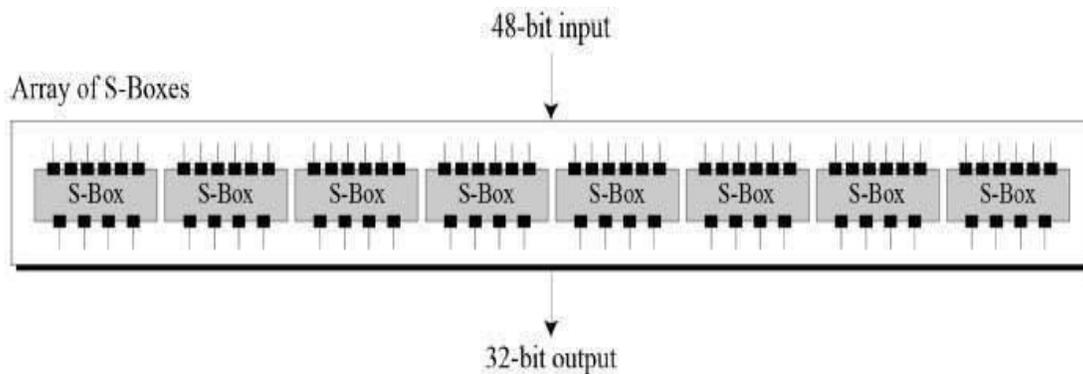


□ The graphically depicted permutation logic is generally described as table in DES

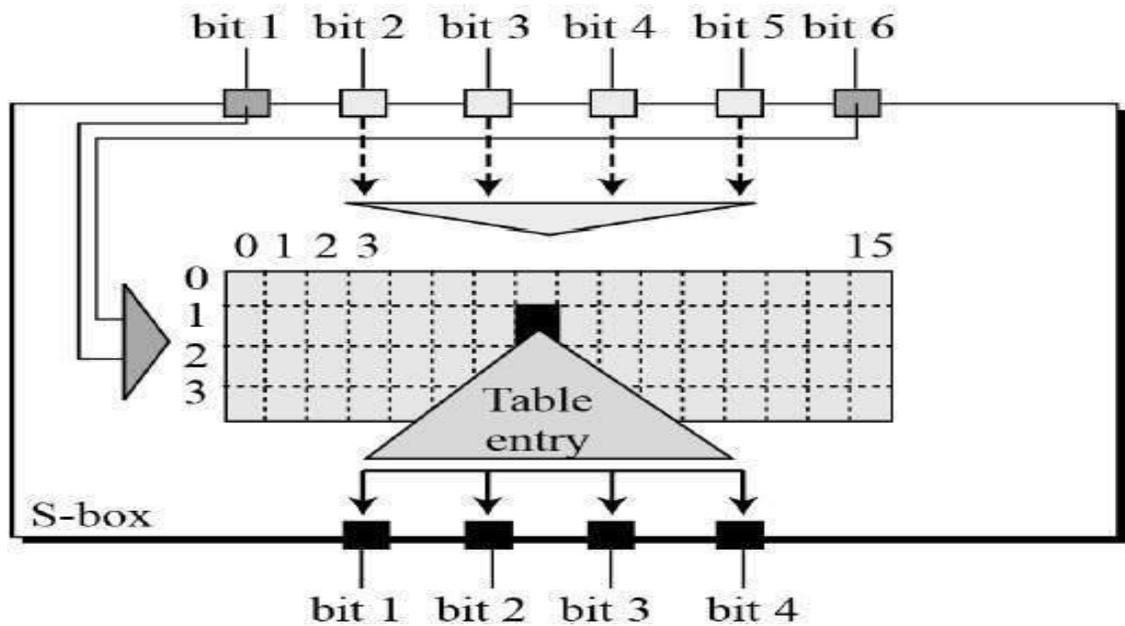
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

specification illustrated as shown –

- **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 –



□ The S-box rule is illustrated below –

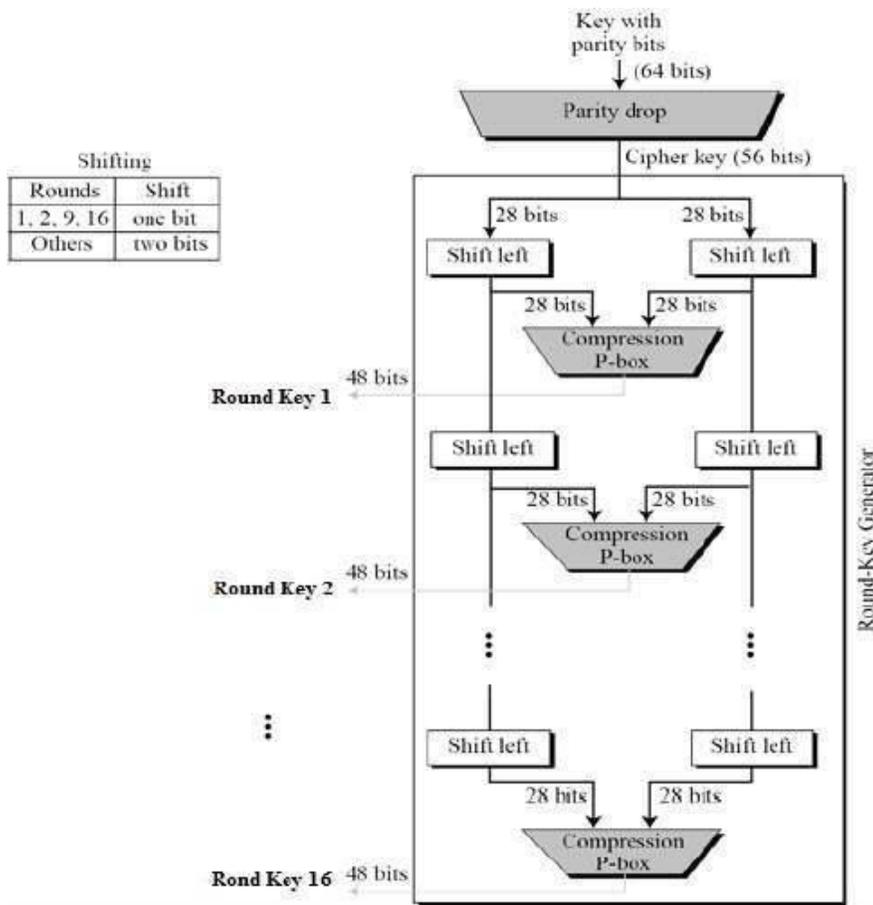


- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

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

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 –



The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

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

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

TRIPLE DES

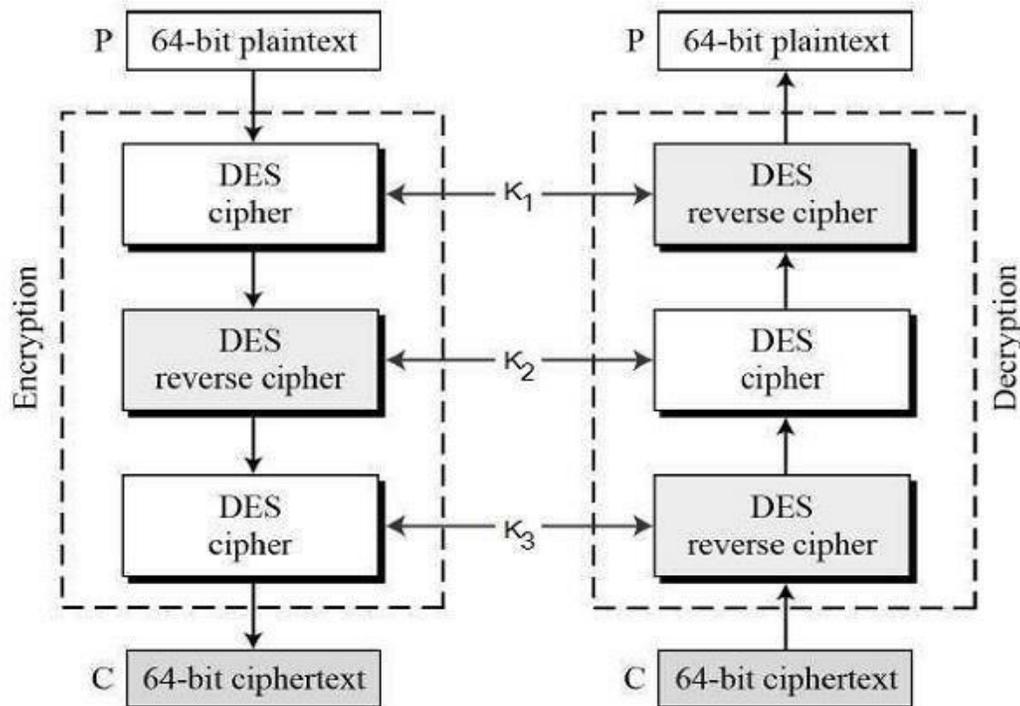
The speed of exhaustive key searches against DES after 1990 began to cause discomfort amongst users of DES. However, users did not want to replace DES as it takes an enormous amount of time and money to change encryption algorithms that are widely adopted and embedded in large security architectures.

The pragmatic approach was not to abandon the DES completely, but to change the manner in which DES is used. This led to the modified schemes of Triple DES (sometimes known as 3DES).

Incidentally, there are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2- key Triple DES (2TDES).

3-KEY Triple DES

Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1 , K_2 and K_3 . This means that the actual 3TDES key has length $3 \times 56 = 168$ bits. The encryption scheme is illustrated as follows –



The encryption-decryption process is as follows –

- Encrypt the plaintext blocks using single DES with key K_1 .
- Now decrypt the output of step 1 using single DES with key K_2 .
- Finally, encrypt the output of step 2 using single DES with key K_3 .
- The output of step 3 is the ciphertext.

- Decryption of a ciphertext is a reverse process. User first decrypt using K3, then encrypt with K2, and finally decrypt with K1.

Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K1, K2, and K3 to be the same value. This provides backwards compatibility with DES.

Second variant of Triple DES (2TDES) is identical to 3TDES except that K3 is replaced by K1. In other words, user encrypt plaintext blocks with key K1, then decrypt with key K2, and finally encrypt with K1 again. Therefore, 2TDES has a key length of 112 bits.

Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.

ADVANCED ENCRYPTION STANDARD

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

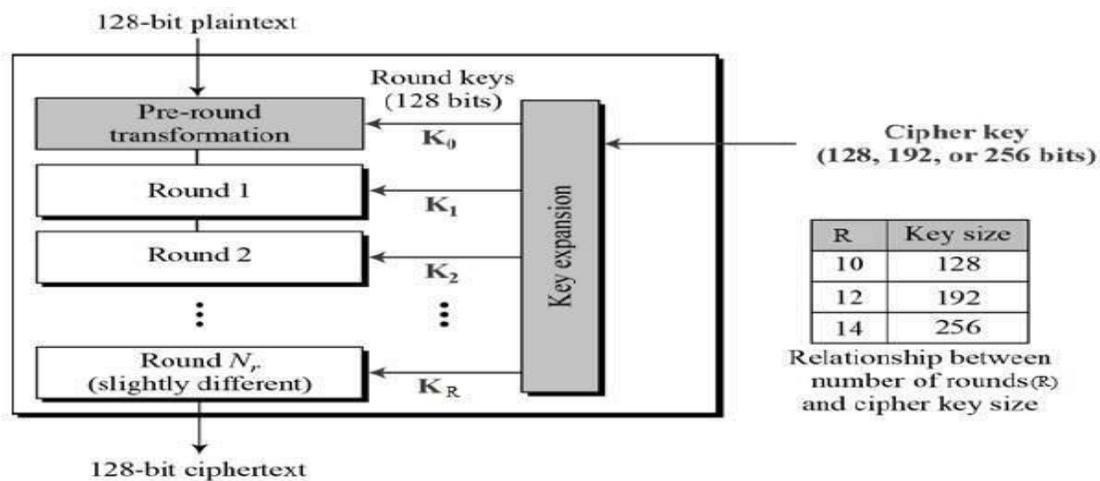
Operation of AES

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 –

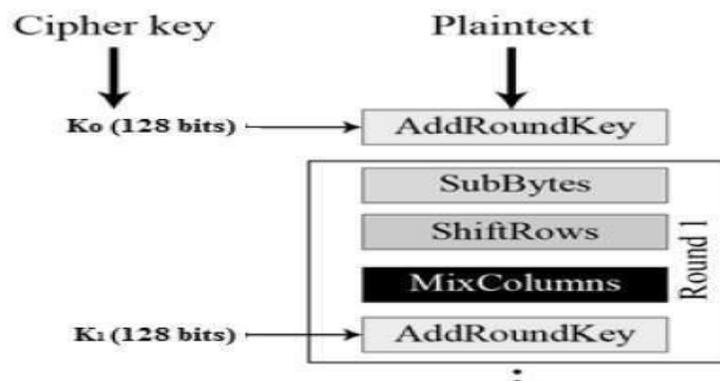
Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration –



Encryption Process

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



Byte Substitution (SubBytes)

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.

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted 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.

MixColumns

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.

Addroundkey

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 ciphertext. 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 ciphertext is similar to the encryption process in the reverse 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 need to be separately implemented, although they are very closely related.

AES Analysis

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES have been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of ‘future-proofing’ against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

BLOCK CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Feistel block cipher. For that reason, it is important to examine the design principles of the Feistel cipher. We begin with a **comparison of stream cipher with block cipher**.

- **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher.
- **A block cipher** is one 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.

Block cipher principles

- Most symmetric block ciphers are based on a **Feistel Cipher Structure** needed since must be able to **decrypt** ciphertext to recover messages efficiently. block ciphers look like an extremely large substitution
- Would need table of 264 entries for a 64-bit block
- Instead create from smaller building blocks
- Using idea of a product cipher in 1949 Claude Shannon introduced idea of substitution-

permutation (S-P) networks called modern substitution-transposition product cipher

- these form the basis of modern block ciphers
- S-P networks are based on the two primitive cryptographic operations we have seen before:
 - *substitution* (S-box)
 - *permutation* (P-box)
- provide *confusion* and *diffusion* of message
 - **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext.
 - **confusion** – makes relationship between ciphertext and key as complex as possible.

BLOCK CIPHER MODES OF OPERATION.

- i. Cipher Block Chaining(CBC)
- ii. Cipher Feedback(CFB)

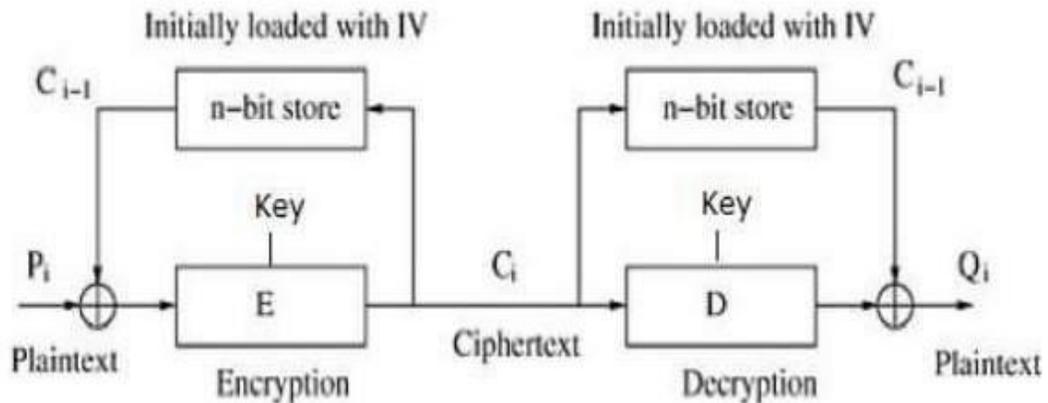
A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

Cipher Block Chaining (CBC) Mode

CBC mode of operation provides message dependence for generating ciphertext and makes the 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 ciphertext block into top register and continue the operation till all plaintext blocks are processed.
- For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.



Analysis of CBC Mode

In CBC mode, the current plaintext block is added to the previous ciphertext block, and then the result is encrypted with the key. Decryption is thus the reverse process, which involves decrypting the current ciphertext and then adding the previous ciphertext block to the result.

Advantage of CBC over ECB is that changing IV results in different ciphertext for identical message. On the drawback side, the error in transmission gets propagated to few further block during decryption due to chaining effect.

It is worth mentioning that CBC mode forms the basis for a well-known data origin authentication mechanism. Thus, it has an advantage for those applications that require both symmetric encryption and data origin authentication.

Cipher Feedback (CFB) Mode

In this mode, each ciphertext block gets ‘fed back’ into the encryption process in order to encrypt 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 ciphertext block.
- Feed ciphertext 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 ciphertext 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.

Analysis of CFB Mode

CFB mode differs significantly from ECB mode, the ciphertext corresponding to a given plaintext block depends not just on that plaintext block and the key, but also on the previous ciphertext block. In other words, the ciphertext block is dependent of message.

CFB has a very strange feature. In this mode, user decrypts the ciphertext using only the encryption process of the block cipher. The decryption algorithm of the underlying block cipher is never used.

Apparently, CFB mode is converting a block cipher into a type of stream cipher. The encryption algorithm is used as a key-stream generator to produce key-stream that is placed in the bottom register. This key stream is then XORed with the plaintext as in case of stream cipher.

By converting a block cipher into a stream cipher, CFB mode provides some of the advantageous properties of a stream cipher while retaining the advantageous properties of a block cipher. On the flip side, the error of transmission gets propagated due to changing of blocks.

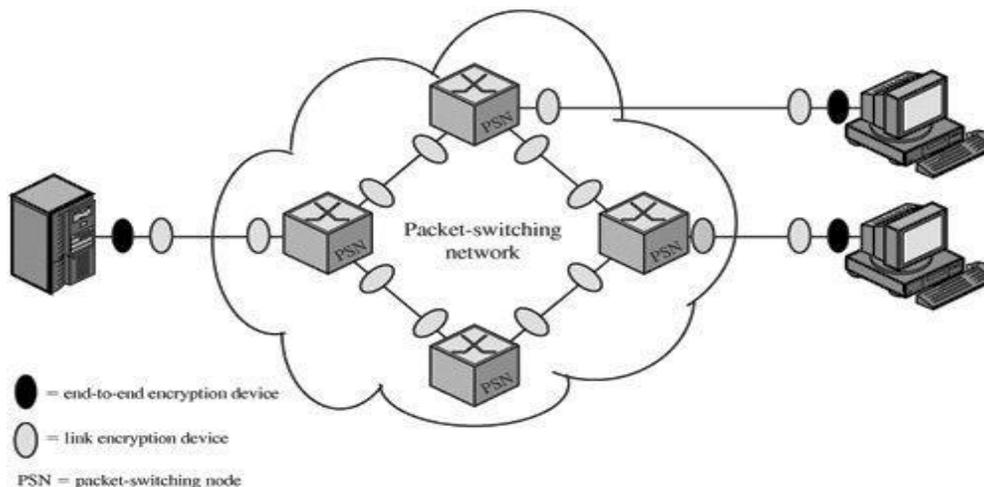
LOCATION OF ENCRYPTION DEVICES

Two approaches are

1. Link encryption
2. End to end encryption

Link versus End-to-End Encryption

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. Although this recourse requires a lot of encryption devices in a large network, its value is clear. One of its disadvantages is that the message must be decrypted each time it enters a switch (such as a frame relay switch) because the switch must read the address (logical connection number) in the packet header in order to route the frame. Thus, the message is vulnerable at each switch. If working with a public network, the user has no control over the security of the nodes. Several implications of link encryption should be noted. For this strategy to be effective, all the potential links in a path from source to destination must use link encryption. Each pair of nodes that share a link should share a unique key, with a different key used on each link. Thus, many keys must be provided.



With end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data. The data in encrypted form are then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data. This plan seems to secure the

transmission against attacks on the network links or switches. Thus, end-to-end encryption relieves the end user of concerns about the degree of security of networks and links that support the communication. There is, however, still a weak spot.

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

Thus, with end-to-end encryption, the user data are secure. However, the traffic pattern is not, because packet headers are transmitted in the clear. On the other hand, end-to-end encryption does provide a degree of authentication. If two end systems share an encryption key, then a recipient is assured that any message that it receives comes from the alleged sender, because only that sender shares the relevant key. Such authentication is not inherent in a link encryption scheme. To achieve greater security, both link and end-to-end encryption are needed. When both forms of encryption are employed, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link encryption key. As the packet traverses the network, each switch decrypts the packet, using a link encryption key to read the header, and then encrypts the entire packet again for sending it out on the next link. Now the entire packet is secure except for the time that the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

Link Encryption	Link Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending host	Message encrypted in sending host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	

Applied by sending host	Applied by sending process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done in hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Key Distribution

For symmetrical encryption systems to work, the two parties must have the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an opponent, Oscar, leaves the key. Therefore, the strength of any cryptographic system rests with the key distribution technique. That is, the means of delivering a key to two parties that wish to exchange data securely.

Key distribution can be achieved in a number of ways. For two parties, Alice and Bob:

- A key could be selected by Alice and physically delivered to Bob:
- A 3rd party could select the key and physically deliver it to Alice and Bob.
- If Alice and Bob have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key.
- If Alice and Bob each have an encrypted connection to a trusted 3rd party, named Casey, he could deliver a key on the encrypted links to Alice and Bob.

Note 1 – Option 1 and 2 call for manual delivery of a key. This is seasonal for link encryption. However, for end-to-end encryption, manual delivery is awkward. In a distributed system, any given host on terminal may need to engage in exchange with many other hosts and terminal over time. Thus each device needs a number of keys, supplied dynamically. The problem becomes quite difficult is a wide area distributed system.

Note 2 – Option 3 is a possibility for either link encryption or end-to-end encryption, but if an opponent, Oscar, ever succeeds in gaining access to one key, then all subsequent keys are revealed.

Note 3 – To provide keys for end-to-end encryption, option 4 is preferable.

UNIT –II: PUBLIC KEY CRYPTOGRAPHY AND MESSAGE AUTHENTICATION

MESSAGE AUTHENTICATION AND HASH FUNCTIONS

Authentication Requirements

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

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

AUTHENTICATION FUNCTIONS

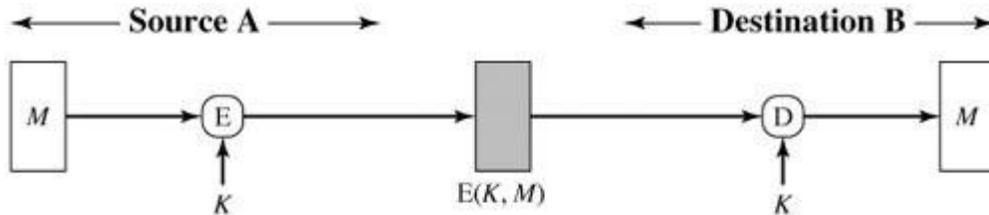
- **Message encryption:** The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- **Hash function:** A function that maps a message of any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

Message Encryption

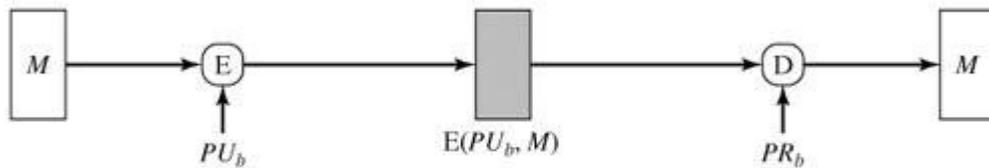
Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

Symmetric Encryption

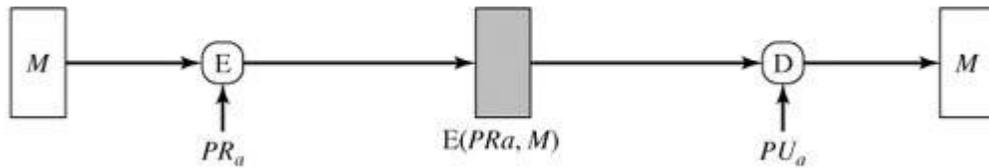
Consider the straightforward use of symmetric encryption in the below figure. A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.



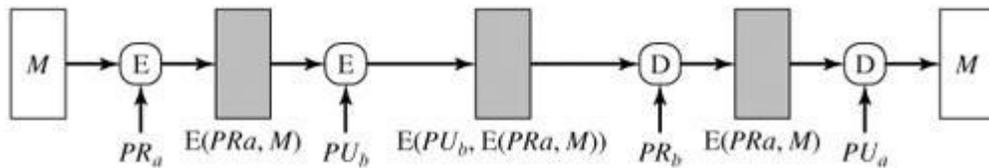
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(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 a cryptographic checksum or MAC that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K . When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\text{MAC} = C(K, M),$$

where

M = input message

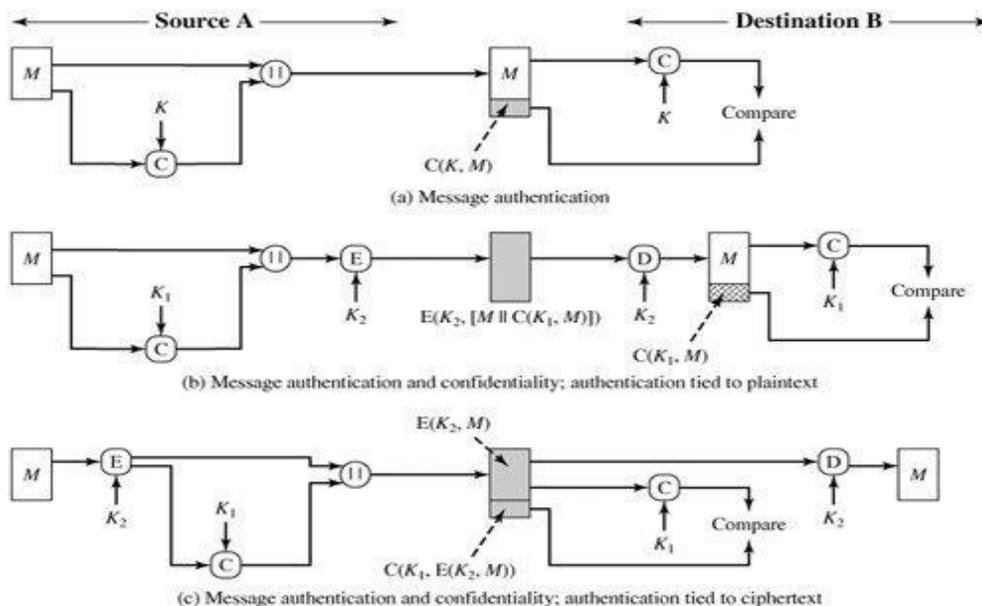
C = MAC function

K = shared secret key

MAC = message authentication code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC. If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

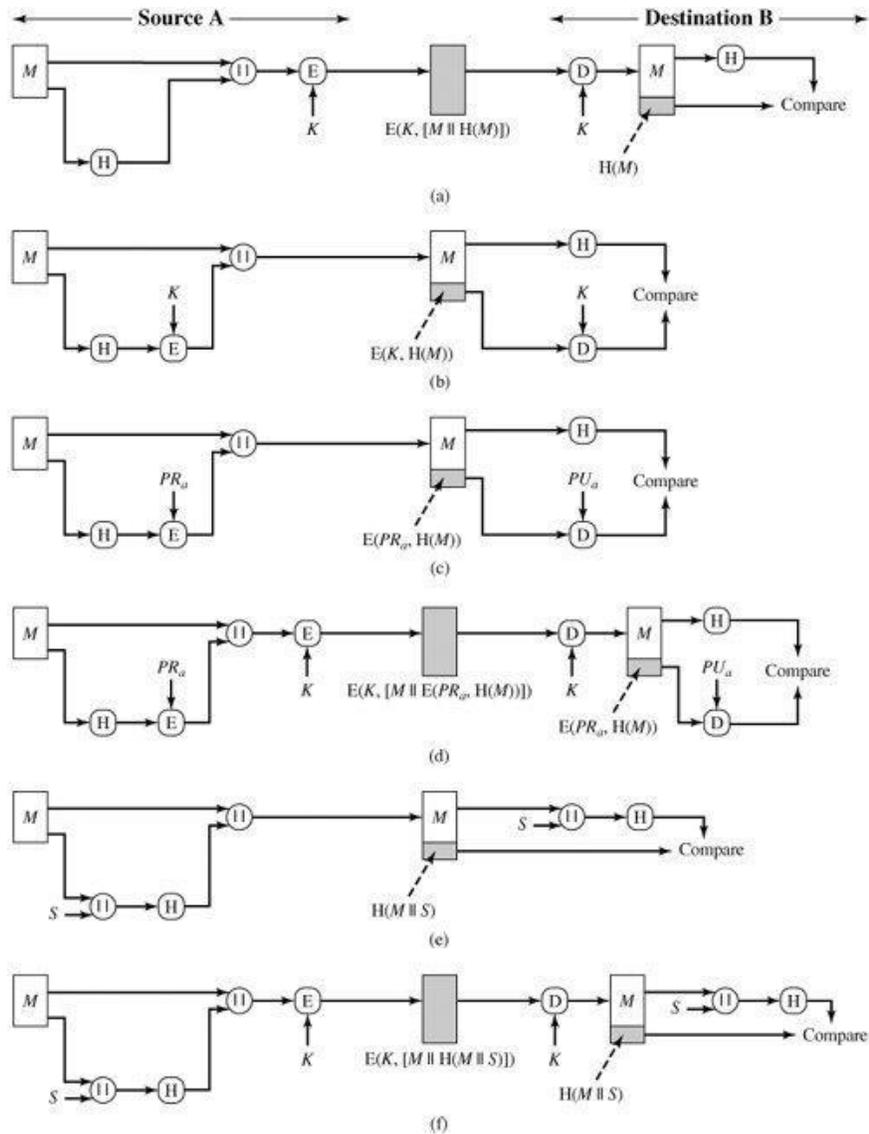
- The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
- The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
- If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.



HASH FUNCTION

A variation on the message authentication code is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed size output, referred to as a **hash code** $H(M)$. Unlike a MAC, a hash code

does not use a key but is a function only of the input message. The hash code is also referred to as a **message digest** or **hash value**. The hash code is a function of all the 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.



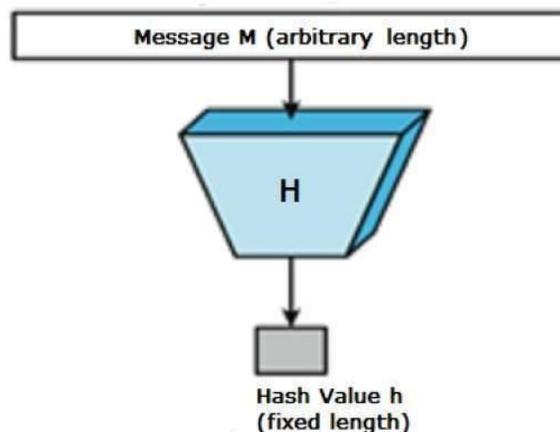
Basic Uses of Hash Function

- The message plus concatenated hash code is encrypted using symmetric encryption. A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

- Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.
- Only the hash code is encrypted, using public-key encryption and using the sender's private key. As with (b), this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.
- If confidentiality as well as a digital signature is desired, then the message plus the private-key encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.
- It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S . A computes the hash value over the concatenation of M and S and appends the resulting hash value to M . Because B possesses S , it can re-compute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.
- Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

Hash functions are extremely useful and appear in almost all information security applications. A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length. Values returned by a hash function are called **message digest** or simply **hash values**. The following picture illustrated hash function.

Features of Hash Functions



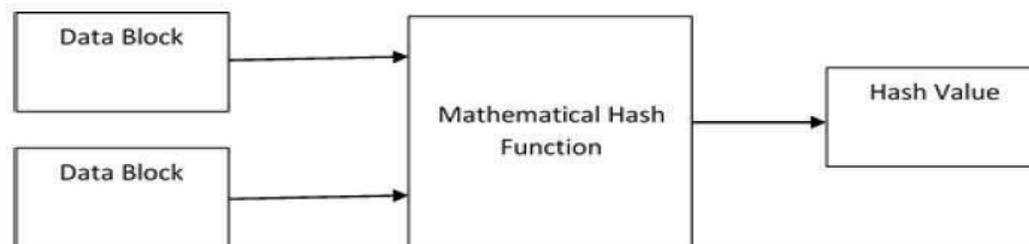
The typical features of hash functions are

- **Fixed Length Output (Hash Value)**
 - Hash function converts data of arbitrary length to a fixed length. This process is often referred to as **hashing the data**.
 - In general, the hash is much smaller than the input data, hence hash functions are sometimes called **compression functions**.
 - Since a hash is a smaller representation of a larger data, it is also referred to as a **digest**.
 - Hash function with n bit output is referred to as an **n-bit hash function**. Popular hash functions generate values between 160 and 512 bits.
- **Efficiency of Operation**
 - Generally for any hash function h with input x , computation of $h(x)$ is a fast operation.
 - Computationally hash functions are much faster than a symmetric encryption.

Design of Hashing Algorithms

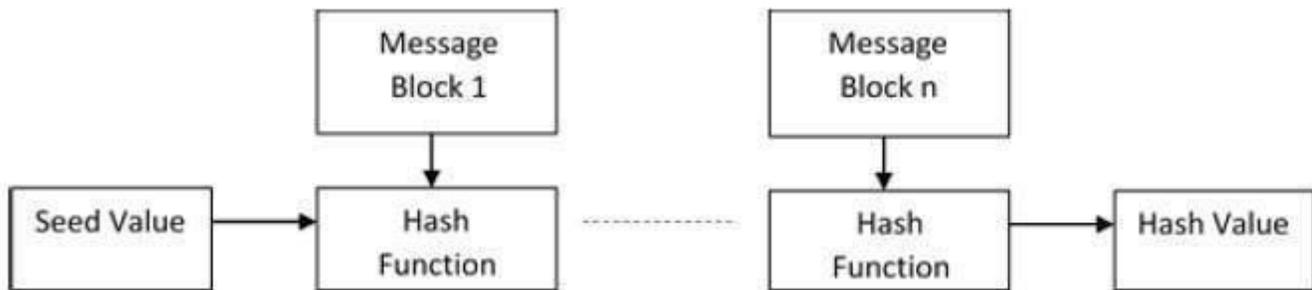
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

The size of each data block varies depending on the algorithm. Typically the block sizes are from 128 bits to 512 bits. The following illustration demonstrates hash function –



Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for as many rounds as are required to hash the entire message. Schematic of hashing algorithm is depicted in the following illustration –



Since, the hash value of first message block becomes an input to the second hash operation, output of which alters the result of the third operation, and so on. This effect, known as an **avalanche** effect of hashing. Avalanche effect results in substantially different hash values for two messages that differ by even a single bit of data. Understand the difference between hash function and algorithm correctly. The hash function generates a hash code by operating on two blocks of fixed-length binary data. Hashing algorithm is a process for using the hash function, specifying how the message will be broken up and how the results from previous message blocks are chained together.

Secure Hash Function (SHA)

Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.

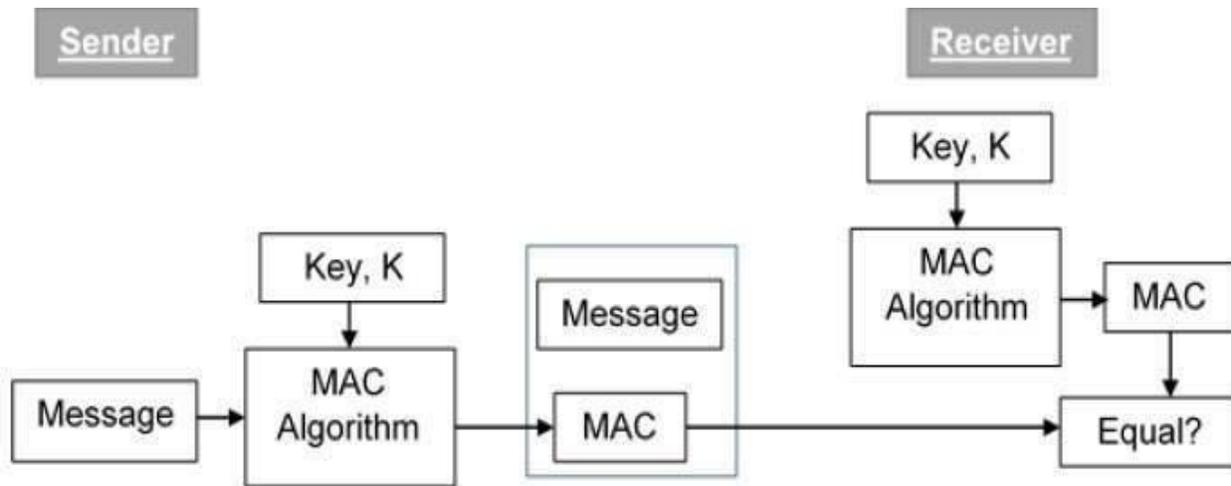
- The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.
- SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.
- In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful.
- SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA- 512 depending up on number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function.
- Though SHA-2 is a strong hash function. Though significantly different, its basic design is still follows design of SHA-1. Hence, NIST called for new competitive hash function designs.
- In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and good resistance for attacks.

Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K .

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail –

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Limitations of MAC

There are two major limitations of MAC, both due to its symmetric nature of operation –

- **Establishment of Shared Secret.**
 - It can provide message authentication among pre-decided legitimate users who have shared key.
 - This requires establishment of shared secret prior to use of MAC.
- **Inability to Provide Non-Repudiation**
 - Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
 - MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.
 - Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

HMAC Algorithm Notations

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
IV = initial value input to hash function

M = message input to HMAC (including the padding specified in the embedded hash function)

Y_i = i th block of M , $0 \leq i < (L - 1)$

L = number of blocks in M

b = number of bits in a block

n = length of hash code produced by embedded hash function

K = secret key recommended length is n ; if key length is greater than b ; the key is input to the hash function to produce an n -bit key

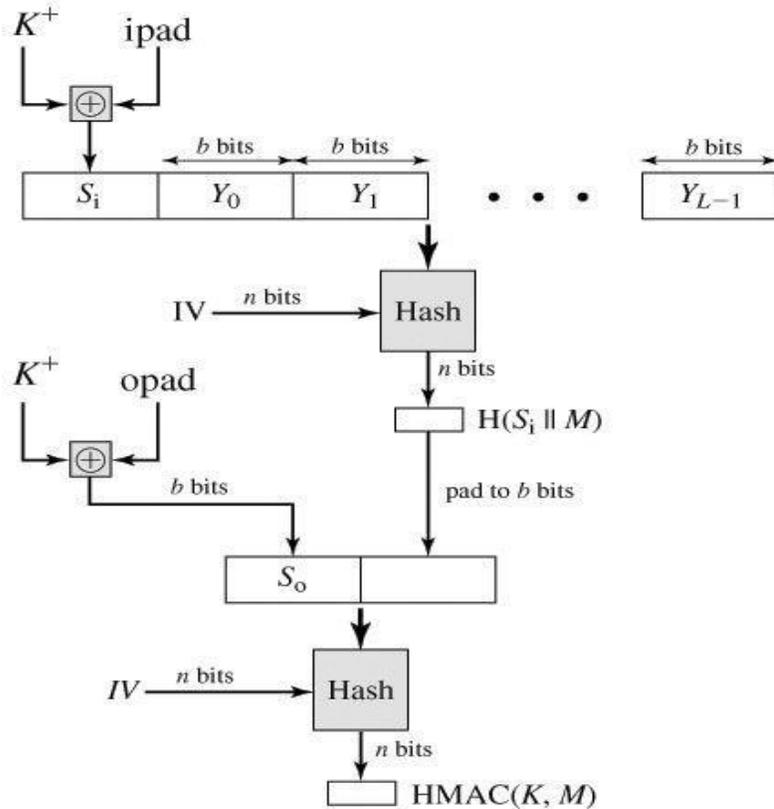
K_+ = K padded with zeros on the left so that the result is b bits in length
ipad = 00110110 (36 in hexadecimal) repeated

$b/8$ times
opad = 01011100 (5C in hexadecimal) repeated

$b/8$ times
Then HMAC can be expressed as follows:

$HMAC(K, M) = H[(K + \text{opad}) || H[(K + \text{ipad}) || M]]$ In words,

1. Append zeros to the left end of K to create a b -bit string K_+ (e.g., if K is of length 160 bits and $b = 512$ then K will be appended with 44 zero bytes 0×00).



2. XOR (bitwise exclusive-OR) K^+ with $ipad$ to produce the b -bit block S_i .
3. Append M to S_i .
4. Apply H to the stream generated in step 3.
5. XOR K^+ with $opad$ to produce the b -bit block S_o
6. Append the hash result from step 4 to S_o
7. Apply H to the stream generated in step 6 and output the result.

Authentication Protocols

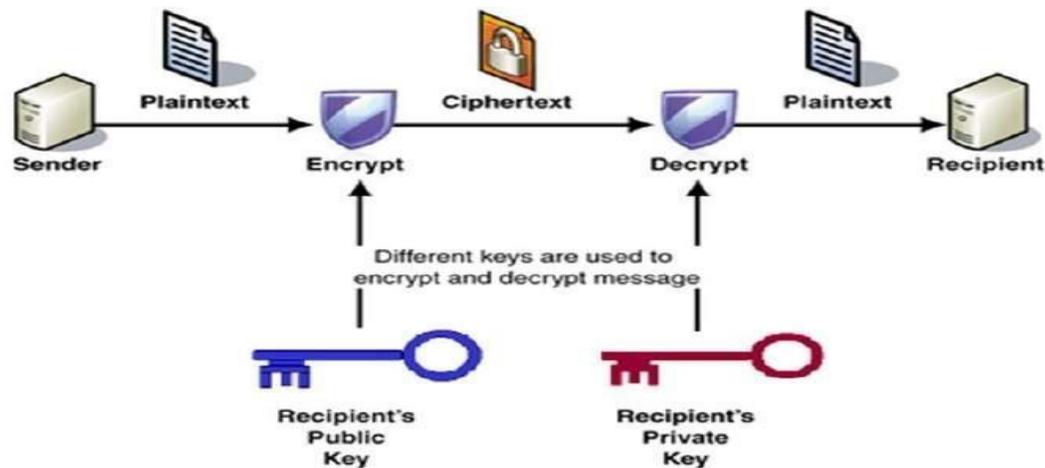
An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

- Authentication Protocols(ap1.o) - Simple authentication Protocol.
- Authentication Protocols(ap2.o) - Network address(IP address)
- Authentication Protocols(ap3.o) - Secret password
- Authentication Protocols(ap3.1) - Encrypted Secret password
- Authentication Protocols(ap4.o) – Nonce(Different password each time)
- Authentication Protocols(ap5.o) – Nonce(Asymmetric key encryption).

Public Key Cryptography Principles

Unlike symmetric key cryptography, we do not find historical use of public-key cryptography. It is a relatively new concept. Symmetric cryptography was well suited for organizations such as governments, military, and big financial corporations were involved in the classified communication. With the spread of more unsecure computer networks in last few decades, a genuine need was felt to use cryptography at larger scale. The symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to the public key cryptosystems.

The process of encryption and decryption is depicted in the following illustration –



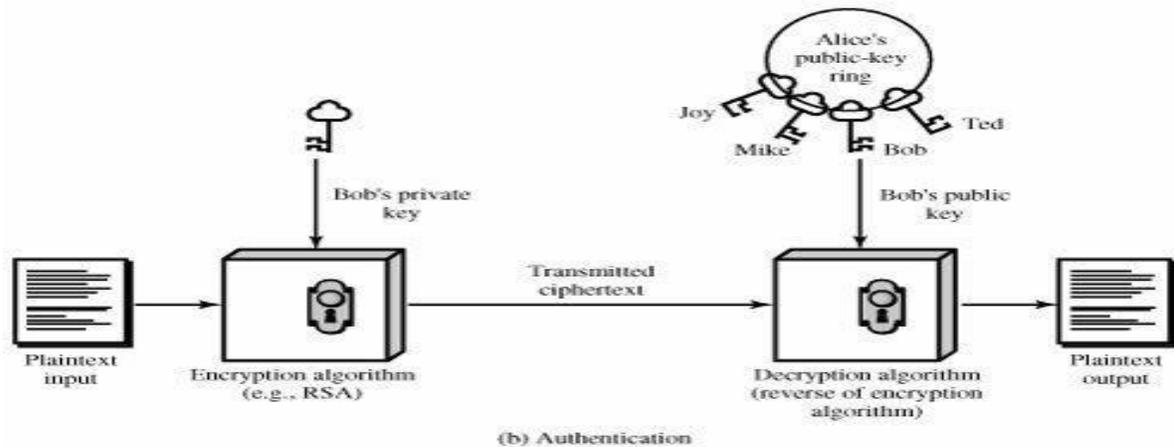
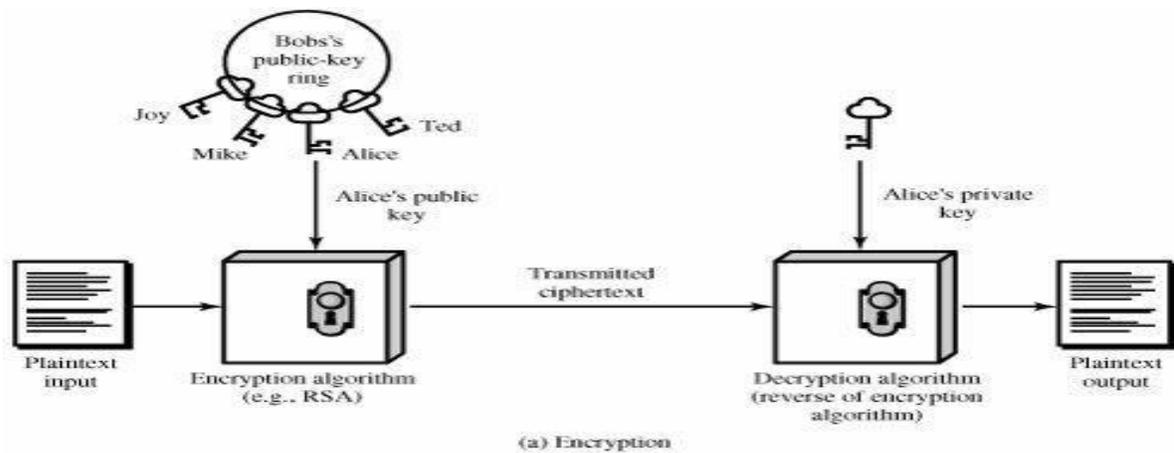
The most important properties of public key encryption scheme are –

- Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
- Each receiver possesses a unique decryption key, generally referred to as his private key.
- Receiver needs to publish an encryption key, referred to as his public key.
- Some assurance of the authenticity of a public key is needed in this scheme to avoid spoofing by adversary as the receiver. Generally, this type of cryptosystem involves trusted third party which certifies that a particular public key belongs to a specific person or entity only.
- Encryption algorithm is complex enough to prohibit attacker from deducing the plaintext from the ciphertext and the encryption (public) key.

Though private and public keys are related mathematically, it is not be feasible to calculate the private key from the public key. In fact, intelligent part of any public-key cryptosystem is in designing a relationship between two keys.

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 algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.



RSA Algorithm

This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars **Ron Rivest, Adi Shamir, and Len Adleman** and hence, it is termed as RSA cryptosystem.

We will see two aspects of the RSA cryptosystem, firstly generation of key pair and secondly encryption-decryption algorithms.

Generation of RSA Key Pair

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below –

- **Generate the RSA modulus (n)**
 - Select two large primes, p and q.
 - Calculate $n=p*q$. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.
- **Find Derived Number (e)**
 - Number e must be greater than 1 and less than $(p - 1)(q - 1)$.
 - There must be no common factor for e and $(p - 1)(q - 1)$ except for 1. In other words two numbers e and $(p - 1)(q - 1)$ are co prime.
- **Form the public key**
 - The pair of numbers (n, e) form the RSA public key and is made public.
 - Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.
- **Generate the private key**
 - Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
 - Number d is the inverse of e modulo $(p - 1)(q - 1)$. This means that d is the number less than $(p - 1)(q - 1)$ such that when multiplied by e, it is equal to 1 modulo $(p - 1)(q - 1)$.
 - This relationship is written mathematically as follows –

$$ed = 1 \text{ mod } (p - 1)(q - 1)$$

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

Example

An example of generating RSA Key pair is given below. (For ease of understanding, the primes p & q taken here are small values. Practically, these values are very high).

- Let two primes be $p = 7$ and $q = 13$. Thus, modulus $n = pq = 7 \times 13 = 91$.
- Select $e = 5$, which is a valid choice since there is no number that is common factor of 5 and $(p - 1)(q - 1) = 6 \times 12 = 72$, except for 1.

- The pair of numbers $(n, e) = (91, 5)$ forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input $p = 7$, $q = 13$, and $e = 5$ to the Extended Euclidean Algorithm. The output will be $d = 29$.
- Check that the d calculated is correct by computing –

$$de = 29 \times 5 = 145 = 1 \pmod{72}$$
- Hence, public key is $(91, 5)$ and private keys is $(91, 29)$.

Encryption and Decryption

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy.

Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n . Hence, it is necessary to represent the plaintext as a series of numbers less than n .

RSA Encryption

- Suppose the sender wish to send some text message to someone whose public key is (n, e) .
- The sender then represents the plaintext as a series of numbers less than n .
- To encrypt the first plaintext P , which is a number modulo n . The encryption process is simple mathematical step as

$$C = P^e \pmod{n}$$

- In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n . This means that C is also a number less than n .
- Returning to our Key Generation example with plaintext $P = 10$, we get ciphertext C

$$C = 10^5 \pmod{91}$$

RSA Decryption

- The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a ciphertext C .
- Receiver raises C to the power of his private key d . The result modulo n will be the plaintext P .

$$\text{Plaintext} = C^d \pmod{n}$$

- Returning again to our numerical example, the ciphertext $C = 82$ would get decrypted to number 10 using private key 29,

$$\text{Plaintext} = 82^{29} \pmod{91} = 10$$

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

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

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod n$

RSA Analysis

The security of RSA depends on the strengths of two separate functions. The RSA cryptosystem is most popular public-key cryptosystem strength of which is based on the practical difficulty of factoring the very large numbers.

- **Encryption Function** – It is considered as a one-way function of converting plaintext into ciphertext and it can be reversed only with the knowledge of private key d .
- **Key Generation** – The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n . An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n . It is also a one way function, going from p & q values to modulus n is easy but reverse is not possible.

The strength of RSA encryption drastically goes down against attacks if the number p and q are not large primes and/ or chosen public key e is a small number.

KEY MANAGEMENT

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

Public Announcement of Public Keys

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large.



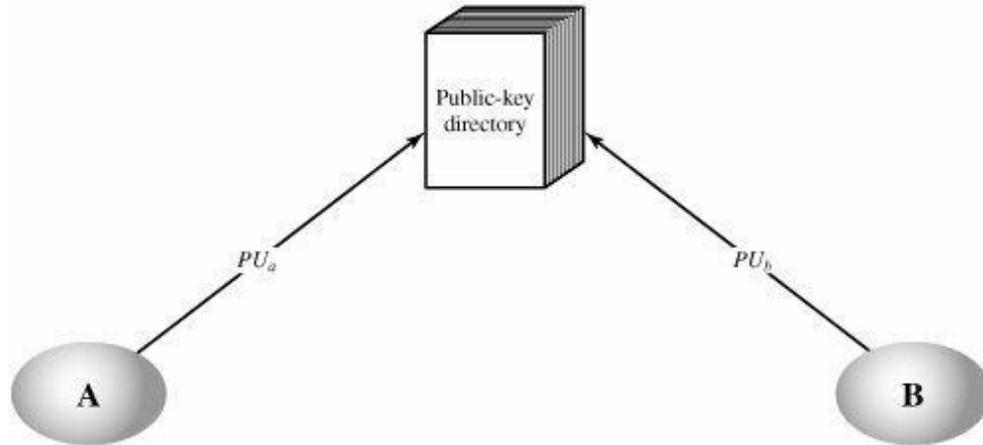
Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants,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

Publicly Available Directory

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

Such a scheme would include the following elements:

- The authority maintains a directory with a {name, public key} entry for each participant.
- Each participant registers a public key with the directory authority. R
- A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
- Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.



This scheme is clearly more secure than individual public announcements but still has vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. The scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. The following steps

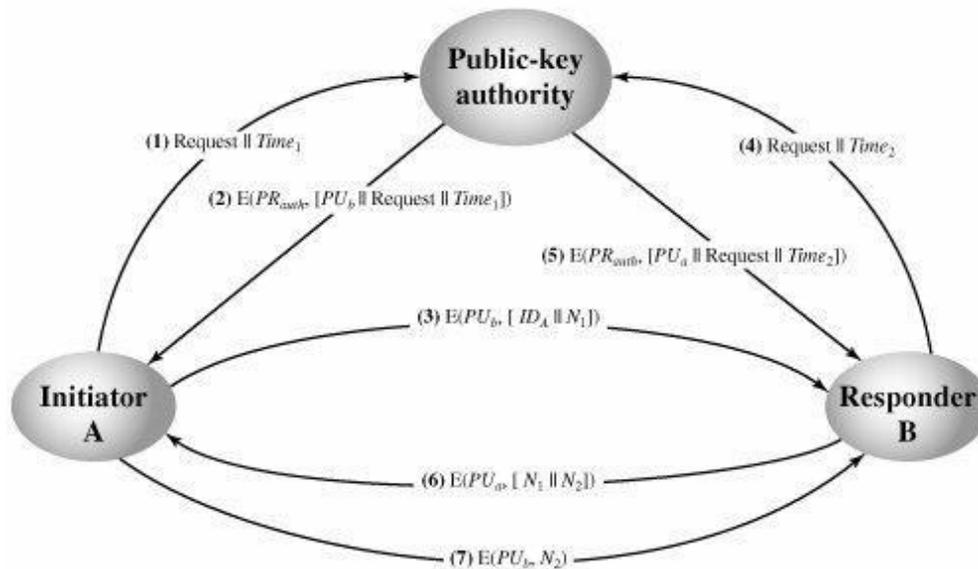
1. A sends a time stamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth}

Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

- B's public key, PU_b which A can use to encrypt messages destined for B
- The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by

the authority.

- The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key.
- 3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (*IDA*) and a nonce (*N1*), which is used to identify this transaction uniquely.
- 4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
- 5. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:
- 6. B sends a message to A encrypted with *PUa* and containing A's nonce (*N1*) as well as a new nonce generated by B (*N2*) Because only B could have decrypted message (3), the presence of *N1* in message (6) assures A that the correspondent is B.
- 7. A returns *N2*, encrypted using B's public key, to assure B that its correspondent is A.



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

Public-Key Certificates

The scenario of Figure 10.3 is attractive, yet it has some drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

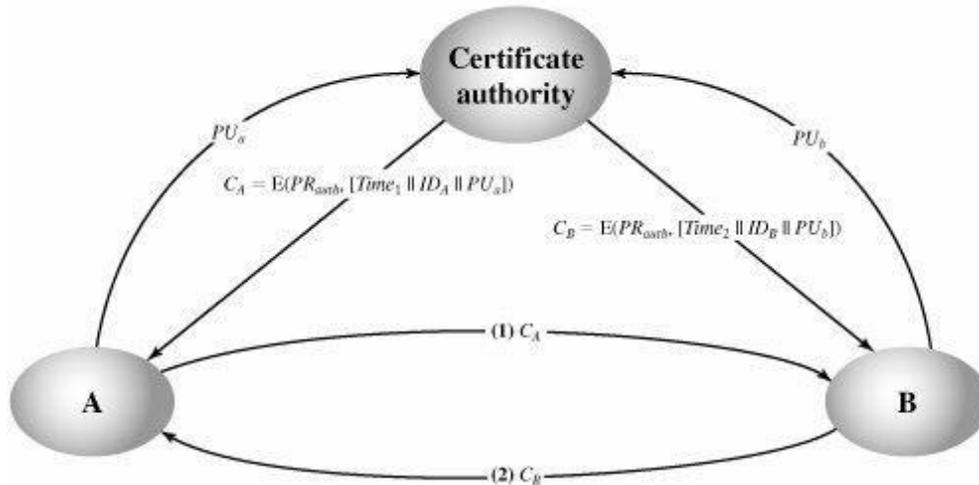
An alternative approach, to use **certificates** that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the

keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate.

Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates
4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.



Application must be in person or by some form of secure authenticated communication.

For participant A, the authority provides a certificate of the form $CA = E(PR_{auth}, [T||IDA||PUa])$ where PR_{auth} is the private key used by the authority and T is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

$$D(PU_{auth}, CA) = D(PU_{auth}, E(PR_{auth}, [T||IDA||PUa])) = (T||IDA||PUa)$$

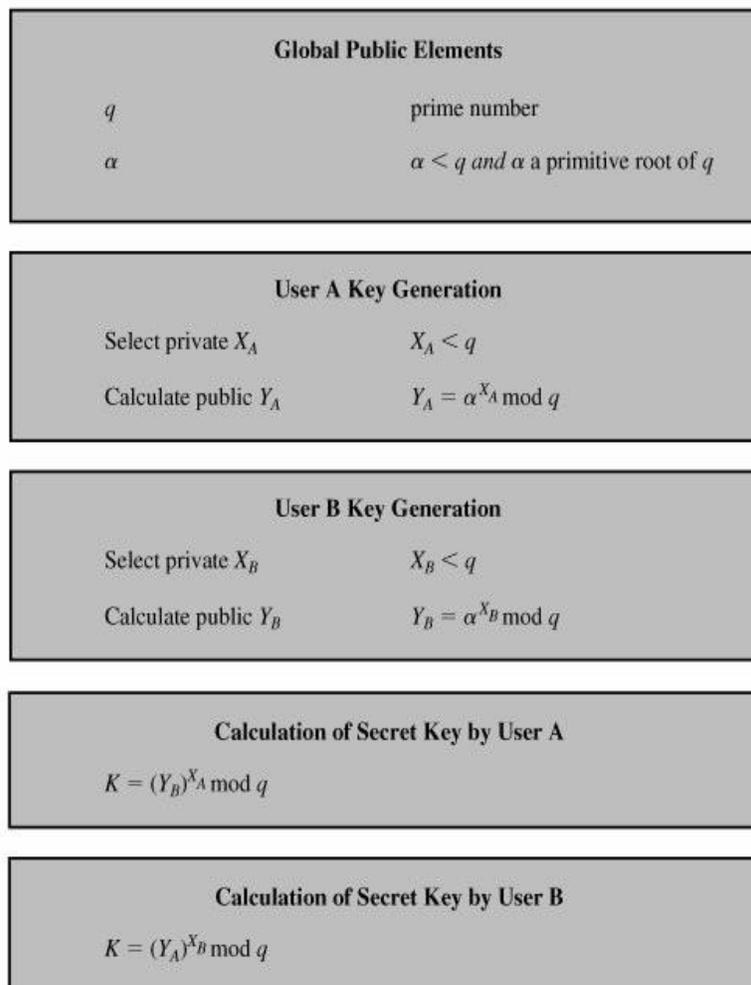
The recipient uses the authority's public key, PU_{auth} to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements IDA and PUa provide the recipient with the

name and public key of the certificate's holder. The timestamp T validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

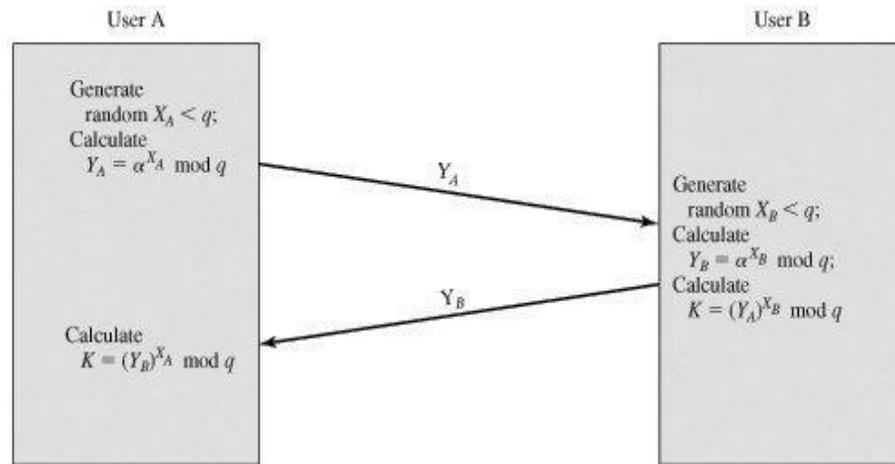
DIFFIE- HELLMAN KEY EXCHANGE

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key exchange. The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values.

- It is not an encryption algorithm
- It Exchange Secret/Symmetric key between end users
- It use Asymmetric encryption



- i. Assume X_A (Private Key of user A) $X_A = q$
- ii. Calculate Y_A (Public key of user A) $Y_A = \alpha^{X_A} \bmod q$
- iii. Assume X_B (Private Key of user B) $X_B = q$
- iv. Calculate Y_B (Public key of user B) $Y_B = \alpha^{X_B} \bmod q$
- v. $K = (Y_B)^{X_A} \bmod q$ for User A
- vi. $K = (Y_A)^{X_B} \bmod q$ for User B



Example

- i. $q=11, \alpha = 2$
- ii. α is a primitive root of p , that is $\alpha \bmod p, \alpha^2 \bmod p, \dots, \alpha^{p-1} \bmod p$ is $1, 2, 3, \dots, p-1$.
- iii. Select $X_A = 8$
 $Y_A = \alpha^{X_A} \bmod q \quad Y_A = 2^8 \bmod 11$
 $Y_A = 3$
- iv. Select $X_B = 4 \quad Y_B = \alpha^{X_B} \bmod q \quad Y_B = 2^4 \bmod 11 \quad Y_B = 5$
- v. For User A
 $K = (Y_B)^{X_A} \bmod q \quad K = (5)^8 \bmod 11$
 $K = 4$
- vi. For User B
 $K = (Y_A)^{X_B} \bmod q \quad K = (3)^4 \bmod 11$
 $K = 4$

User A and B have a same Key values.

ElGamal Cryptosystem

Along with RSA, there are other public-key cryptosystems proposed. Many of them are based on different versions of the Discrete Logarithm Problem.

ElGamal cryptosystem, called Elliptic Curve Variant, is based on the Discrete Logarithm Problem. It derives the strength from the assumption that the discrete logarithms cannot be found in practical time frame for a given number, while the inverse operation of the power can be computed efficiently.

Let us go through a simple version of ElGamal that works with numbers modulo p . In the case of elliptic curve variants, it is based on quite different number systems.

Generation of ElGamal Key Pair

Each user of ElGamal cryptosystem generates the key pair through as follows –

- **Choosing a large prime p .** Generally a prime number of 1024 to 2048 bits length is chosen.
- **Choosing a generator element g .**
 - This number must be between 1 and $p - 1$, but cannot be any number.
 - It is a generator of the multiplicative group of integers modulo p . This means for every integer m co-prime to p , there is an integer k such that $g^k = a \pmod{p}$. For example, 3 is generator of group 5 ($Z_5 = \{1, 2, 3, 4\}$).

N	3^n	$3^n \pmod{5}$
1	3	3
2	9	4
3	27	2
4	81	1

- **Choosing the private key.** The private key x is any number bigger than 1 and smaller than $p-1$.
- **Computing part of the public key.** The value y is computed from the parameters p , g and the private key x as follows –

$$y = g^x \pmod{p}$$

- **Obtaining Public key.** The ElGamal public key consists of the three parameters (p , g , y).

For example, suppose that $p = 17$ and that $g = 6$ (It can be confirmed that 6 is a generator of group Z_{17}). The private key x can be any number bigger than 1 and smaller than 16, so we choose $x = 5$. The value y is then computed as follows –

$$y = 6^5 \pmod{17} = 7$$

- Thus the private key is 5 and the public key is (17, 6, 7).

Encryption and Decryption

The generation of an ElGamal key pair is comparatively simpler than the equivalent process for RSA. But the encryption and decryption are slightly more complex than RSA.

ElGamal Encryption

Suppose sender wishes to send a plaintext to someone whose ElGamal public key is (p, g, y) , then –

- Sender represents the plaintext as a series of numbers modulo p .
- To encrypt the first plaintext P , which is represented as a number modulo p . The encryption process to obtain the ciphertext C is as follows –
 - Randomly generate a number k ;
 - Compute two values $C1$ and $C2$, where –

$$C1 = g^k \text{ mod } p$$
$$C2 = (P * y^k) \text{ mod } p$$

- Send the ciphertext C , consisting of the two separate values $(C1, C2)$, sent together.
- Referring to our ElGamal key generation example given above, the plaintext $P = 13$ is encrypted as follows –
 - Randomly generate a number, say $k = 10$
 - Compute the two values $C1$ and $C2$, where –

$$C1 = 6^{10} \text{ mod } 17$$
$$C2 = (13 * 7^{10}) \text{ mod } 17 = 9$$

- Send the ciphertext $C = (C1, C2) = (15, 9)$.

ElGamal Decryption

- To decrypt the ciphertext $(C1, C2)$ using private key x , the following two steps are taken –
 - Compute the modular inverse of $(C1)^x$ modulo p , which is $(C1)^{-x}$, generally referred to as decryption factor.
 - Obtain the plaintext by using the following formula –

$$C2 \times (C1)^{-x} \text{ mod } p = \text{Plaintext}$$

- In our example, to decrypt the ciphertext $C = (C1, C2) = (15, 9)$ using private key $x = 5$, the decryption factor is

$$15^{-5} \bmod 17 = 9$$

- Extract plaintext $P = (9 \times 9) \bmod 17 = 13$.

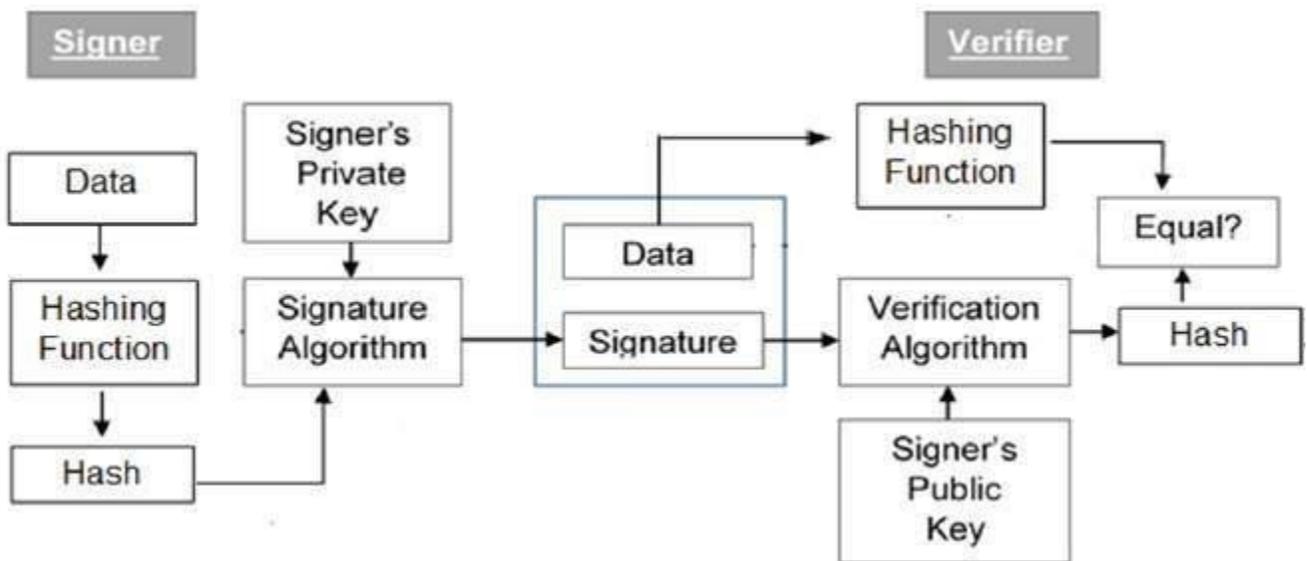
ElGamal Analysis

In ElGamal system, each user has a private key x . and has **three components** of public key – **prime modulus p , generator g , and public $Y = g^x \bmod p$** . The strength of the ElGamal is based on the difficulty of discrete logarithm problem.

The secure key size is generally > 1024 bits. Today even 2048 bits long key are used. On the processing speed front, Elgamal is quite slow, it is used mainly for key authentication protocols. Due to higher processing efficiency, Elliptic Curve variants of ElGamal are becoming increasingly popular.

DIGITAL SIGNATURE

The digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration –



The following points explain the entire process in detail –

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification

algorithm. The verification algorithm gives some value as output.

- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by ‘private’ key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data**.

Importance of Digital Signature

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security. Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature

- **Message authentication** – When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- **Data Integrity** – In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- **Non-repudiation** – Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

Key Management

Key management refers to the distribution of cryptographic keys; the mechanisms used to bind an identity to a key; and the generation, maintenance, and revoking of such keys.

The notation that we will use is

$$X \rightarrow Y: \{ Z \} k$$

means that entity X sends to entity Y a message Z encrypted with the key k

e.g.

Alice \rightarrow Bob: { "Hello World" } k

means that Alice send Bob the message "Hello World" using key k. k represents the secret key for the classical (symmetrical) key encryption system. e and d represent the public and private key, respectively, for a public key (asymmetrical) encryption system.

Session and Interchange Keys

Def: An *interchange key* is a cryptographic key associated with a principal to a communication.

Def: A *session key* is a cryptographic key associated with the communications itself talk about way to communicate different key for each communications A session key prevents forward searches. Forward Search Attack small number of plain text messages encrypt with a public key compare to sent messages know plain text message.

e.g.

Suppose that Alice is a client of Bob's stock brokerage firm. Alice need to send Bob one of two messages: BUY or SELL. Cathy, the attacker, enciphers both messages with Bob's public key. When Alice sends her message, Cathy compares it with her enciphered messages and sees which one it matches.

Randomly generated session key that are used once prevents this type of attack. An interchange key used to convince receiver who the sender is used for all session's changes independently of session initiation and termination

UNIT –III: ELECTRONIC MAIL SECURITY

Electronic Mail Security

Two Schemes are

- i. PGP
- ii. S/MIME

PGP (Pretty Good Privacy)

It provide five type of services

- i. Authentication
- ii. Confidentiality
- iii. Compression
- iv. E-mail Compatibility
- v. S

egmentation

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

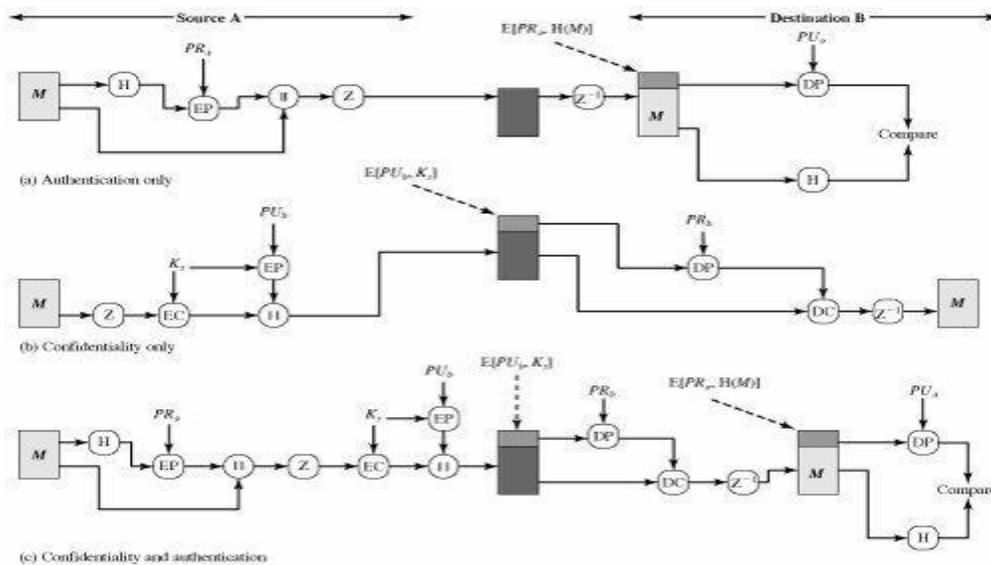
Z = compression using ZIP algorithm

R64 = conversion to radix 64 ASCII format

Authentication

- 1.The sender creates a message.

2. SHA-1 is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.



Confidentiality

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

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.

E-mail Compatibility

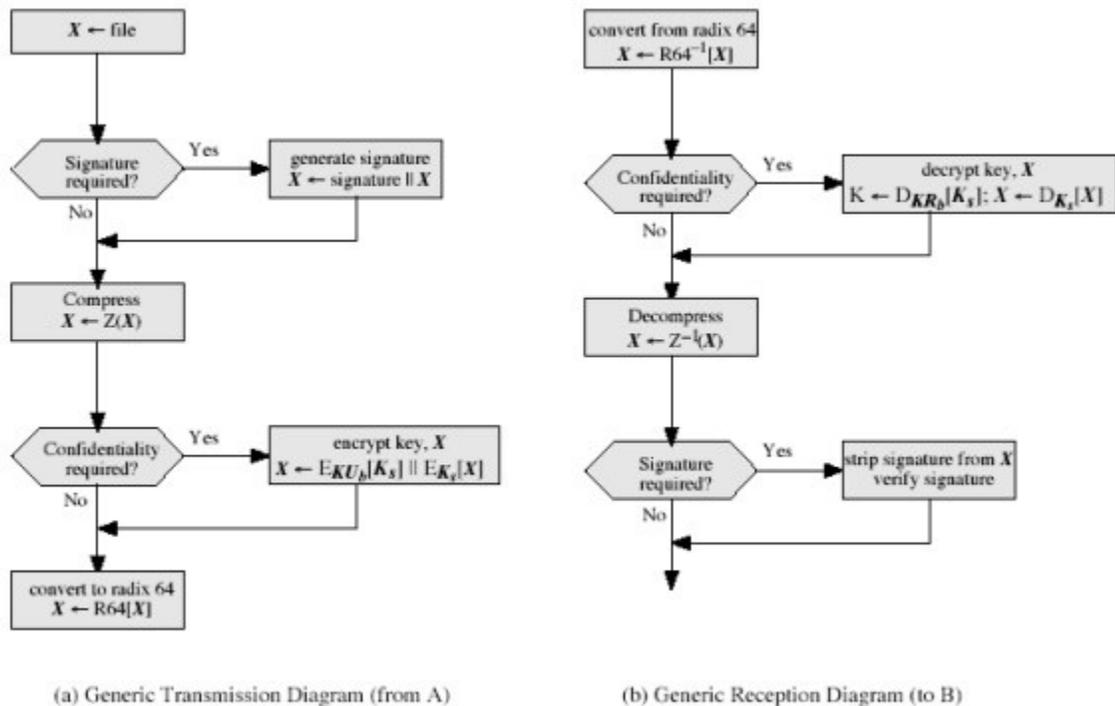
When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters.

Segmentation and 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. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing.

PGP Operation Summary



Cryptographic keys and key rings

Three separate requirements can be identified with respect to these keys: A means of generating unpredictable session keys is needed.

It must allow a user to have multiple public key/private key pairs.

- Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

We now examine each of the requirements in turn.

1. Session key generation

Each session key is associated with a single message and is used only for the purpose of encryption and decryption of that message. Random 128-bit numbers are generated using CAST-128 itself. The

input to the random number generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key. The plaintext input to CAST-128 is itself derived from a stream of 128-bit randomized numbers. These numbers are based on the keystroke input from the user.

2. Key identifiers

If multiple public/private key pair are used, then how does the recipient know which of the public keys was used to encrypt the session key? One simple solution would be to transmit the public key with the message but, it is unnecessary wasteful of space. Another solution would be to associate an identifier with each public key that is unique at least within each user.

The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID. The key ID associated with each public key consists of its least significant 64 bits. i.e., the key ID of public key KU_a is

$$(KU_a \bmod 2^{64})$$

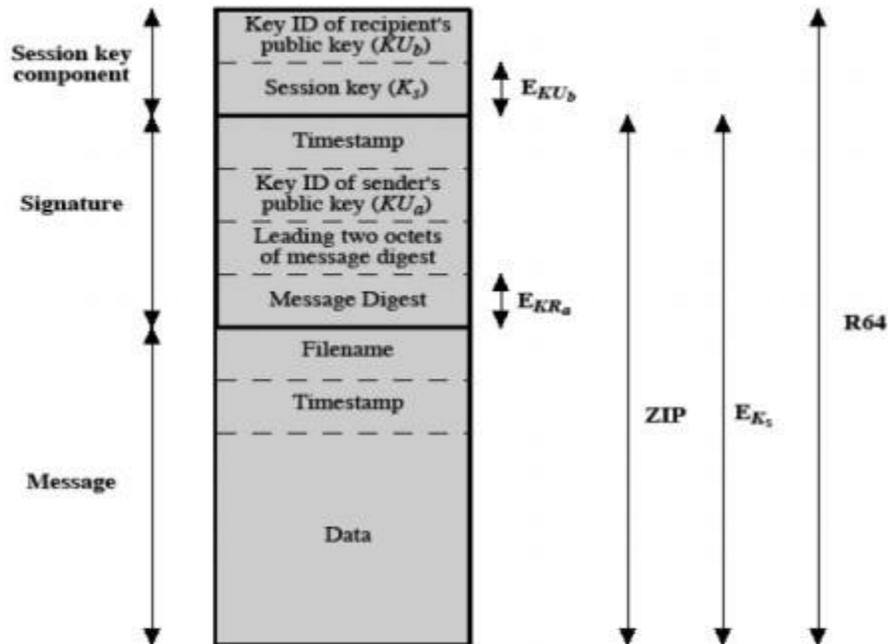
Message consists of three components

- **Message component** – includes actual data to be transmitted, as well as the filename and a timestamp that specifies the time of creation.
- **Signature component** – includes the following
- **Timestamp** – time at which the signature was made. Message digest – hash code.

Two octets of message digest – to enable the recipient to determine if the correct public key was used to decrypt the message.

Key ID of sender's public key – identifies the public key

Session key component – includes session key and the identifier of the recipient public key.



Notation:
 E_{KU_b} = encryption with user b's public key
 E_{KR_a} = encryption with user a's private key
 E_{K_s} = encryption with session key
 ZIP = Zip compression function
 R64 = Radix-64 conversion function

Key rings

PGP provides a pair of data structures at each node, one to store the public/private key pair owned by that node and one to store the public keys of the other users known at that node. These data structures are referred to as private key ring and public key ring.

The general structures of the private and public key rings are shown below:

Timestamp – the date/time when this entry was made.

Key ID – the least significant bits of the public key.

Public key – public key portion of the pair.

Private key – private key portion of the pair.

User ID – the owner of the key.

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
.
.
.
T_1	$PU_i \text{ mod } 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
.
.
.

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
.
.
.
T_1	$PU_i \text{ mod } 2^{64}$	PU_i	trust_flag_i	User i	trust_flag_i		
.
.
.

* = field used to index table

The above figures indicates the General Structure of Private and Public Rings Signature trust field – indicates the degree to which this PGP user trusts the signer to certify public key.

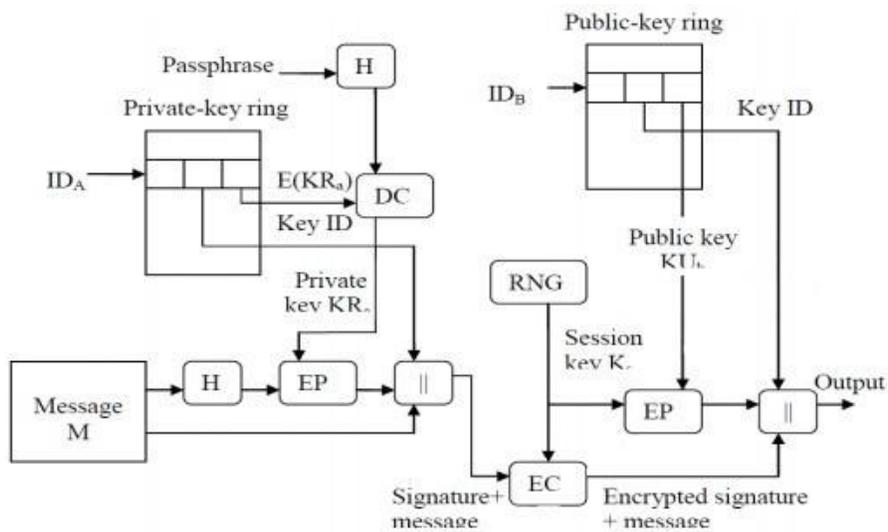
Owner trust field – indicates the degree to which this public key is trusted to sign other public key certificates.

PGP message generation

First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

Signing the message

- PGP retrieves the sender's private key from the private key ring using user ID as an index.
- If user ID was not provided, the first private key from the ring is retrieved.
- PGP prompts the user for the passphrase (password) to recover the unencrypted private key.
- The signature component of the message is constructed.



Encrypting the message

- PGP generates a session key and encrypts the message.
- PGP retrieves the recipient's public key from the public key ring using user ID as index.
- The session key component of the message is constructed

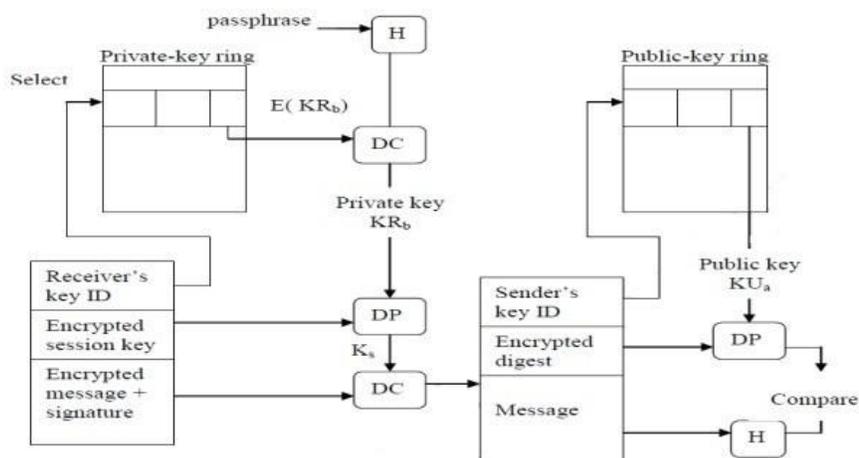


Figure: PGP message reception

Decrypting the message

- PGP retrieves the receiver's private key from the private key ring, using the key ID field in the session key component of the message as an index.
- PGP prompts the user for the passphrase (password) to recover the unencrypted private key.
- PGP then recovers the session key and decrypts the message.

Authenticating the message

- PGP retrieves the sender's public key from the public key ring, using the key ID field in the signature key component of the message as an index.
- PGP recovers the transmitted message digest.
- 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 email format standard, based on technology from RSA Data Security. Although both PGP and S/MIME are on an IETF standards track, it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users.

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 lists the following limitations of the SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/UUdecode

scheme. However, none of these is a standard or even a defacto standard.

2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:
 - Deletion, addition, or reordering of carriage return and linefeed
 - Truncating or wrapping lines longer than 76 characters
 - Removal of trailing white space (tab and space characters)
 - Padding of lines in a message to the same length
 - Conversion of tab characters into multiple space characters

MIME – Header Fields

- 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.

<i>MIME Content Types (This item is displayed on page 461 in the print version)</i>		
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.
Message	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
	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.
Image	External-body	Contains a pointer to an object that exists elsewhere.
	jpeg	The image is in JPEG format, JFIF encoding.
Video	gif	The image is in GIF format.
	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.

<i>MIME Transfer Encodings</i>	
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.

S/MIME Functionality

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability. We then look in more detail at this capability by examining message formats and message preparation.

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

In 1994, the Internet Architecture Board (IAB) issued a report titled “Security in the Internet Architecture” (RFC 1636). The report identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms.

To provide security, the IAB included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. This means that vendors can begin offering these features now, and many vendors now do have some IPsec capability in their products. The IPsec specification now exists as a set of Internet standards.

Applications of IPsec

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

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

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

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

Enhancing electronic commerce security: Even though some Web and electronic commerce applications have built in security protocols, the use of IPsec enhances that security. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer.

The principal feature of IPsec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed applications (including remote logon, client/server, e-mail, file transfer, Web access, and so on) can be secured.

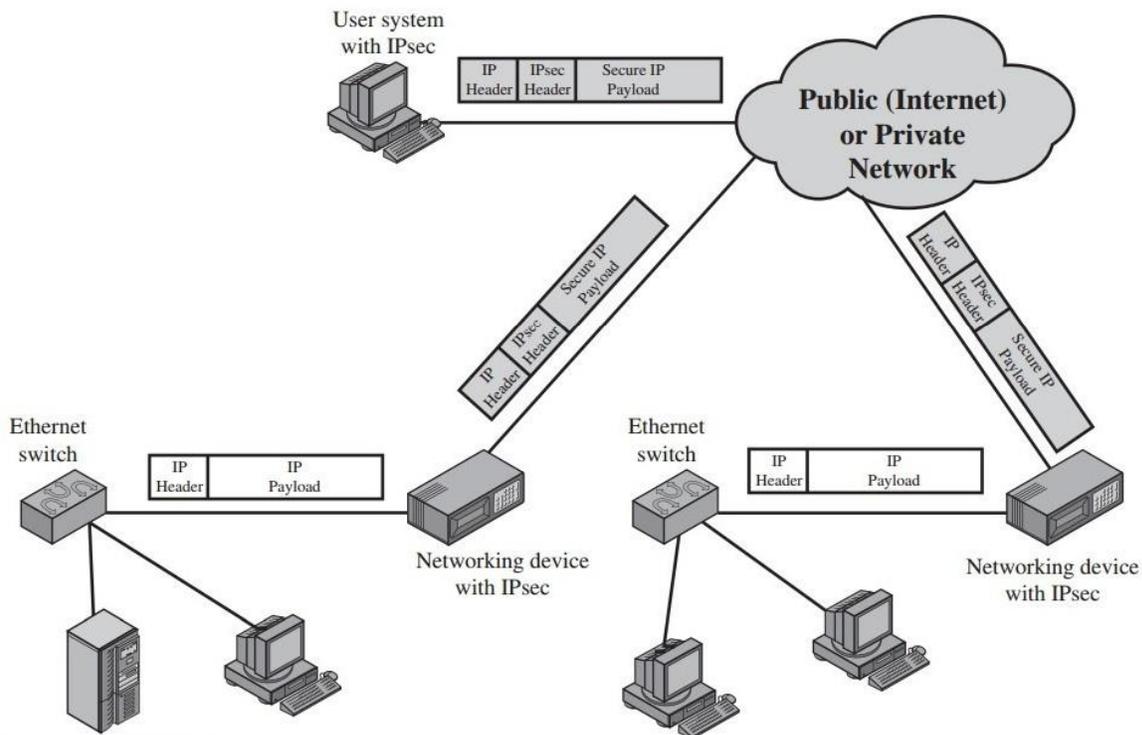


Figure 19.1 An IP Security Scenario

Benefits of IPsec

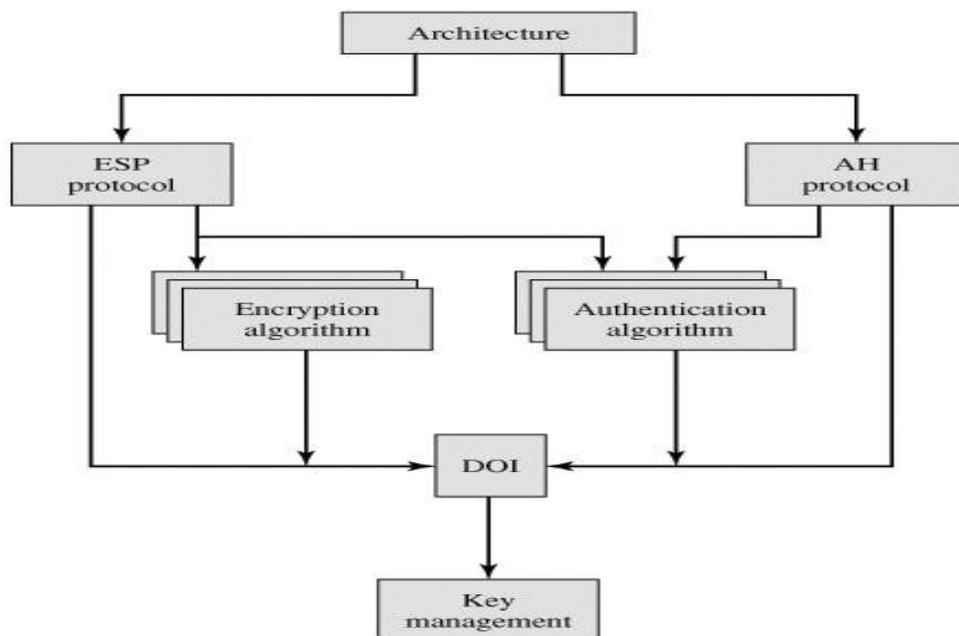
When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

- IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP, and the firewall is the only means of entrance from the Internet into the organization.
- IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router. Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.

- IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

IP Security Architecture

- **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 to the 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.



1. Architecture

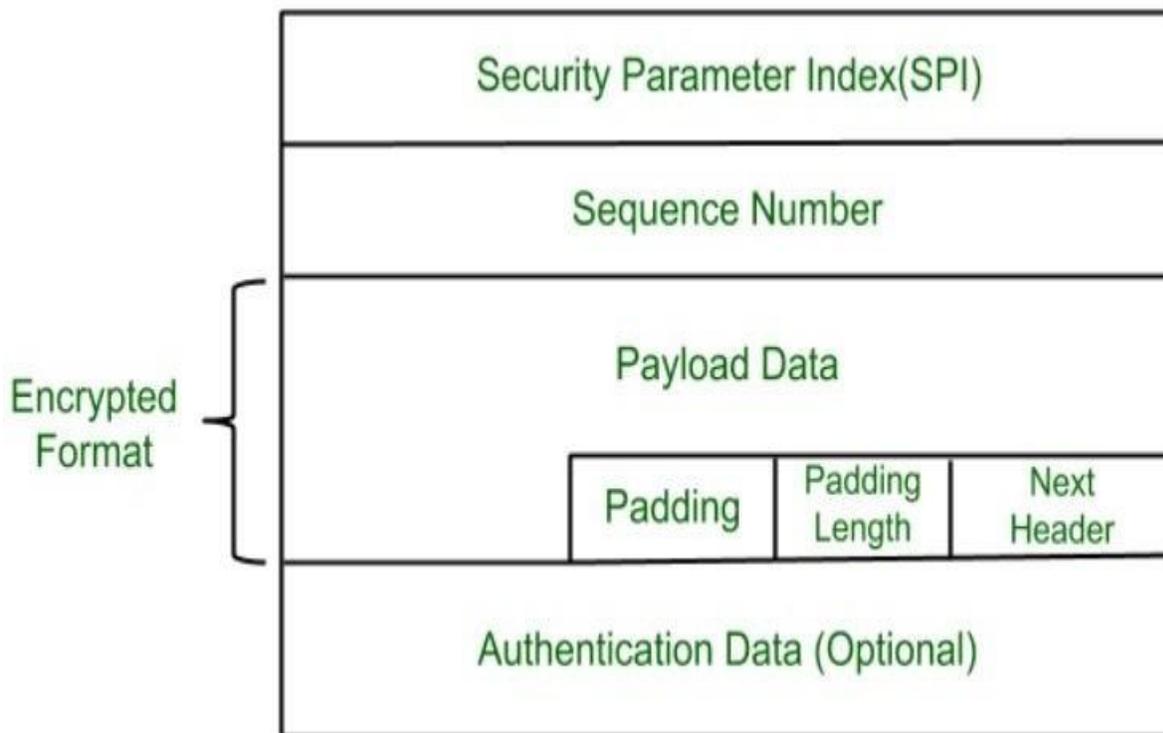
Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.

2. ESP Protocol

ESP(Encapsulation Security Payload) provide the confidentiality service. Encapsulation Security Payload is implemented in either two ways:

- ESP with optional Authentication.
- ESP with Authentication.

Packet Format:



Security Parameter Index(SPI):

This parameter is used in Security Association. It is used to give a unique number to the connection build between Client and Server.

Sequence Number:

Unique Sequence number are allotted to every packet so that at the receiver side packets can be arranged properly.

Payload Data:

Payload data means the actual data or the actual message. The Payload data is in encrypted format to achieve confidentiality.

Padding:

Extra bits or space added to the original message in order to ensure confidentiality. Padding length is the size of the added bits or space in the original message.

Next Header:

Next header means the next payload or next actual data.

Authentication Data

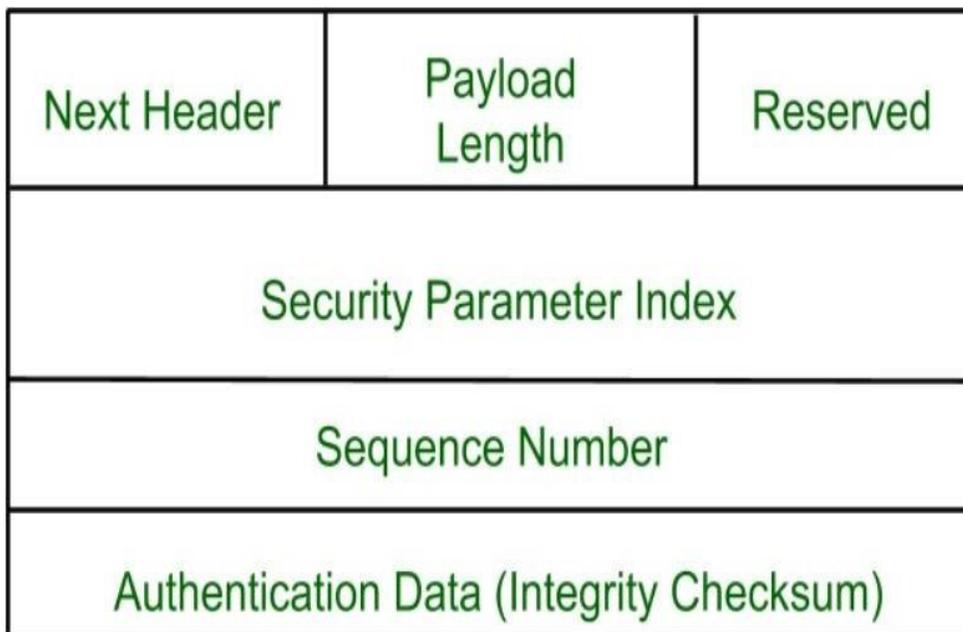
This field is optional in ESP protocol packet format.

3. Encryption algorithm:

Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.

4. AH Protocol:

AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity.



Authentication Header covers the packet format and general issue related to the use of AH for packet authentication and integrity.

5. Authentication Algorithm:

Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.

6. DOI (Domain of Interpretation):

DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.

7. Key Management:

Key Management contains the document that describes how the keys are exchanged between sender and receiver.

Authentication Header (AH):

The Authentication Header (AH) is an IPSec protocol that provides data integrity, data origin authentication, and optional anti-replay services to IP. Authentication Header (AH) does not provide any data confidentiality (Data encryption). Since Authentication Header (AH) does not provide confidentiality, there is no need for an encryption algorithm. AH protocol is specified in RFC 2402.

Authentication Header (AH) is an IP protocol and has been assigned the protocol number 51 by IANA. In the IP header of Authentication Header (AH) protected datagram, the 8-bit protocol field will be 51, indicating that following the IP header is an Authentication Header (AH) header.

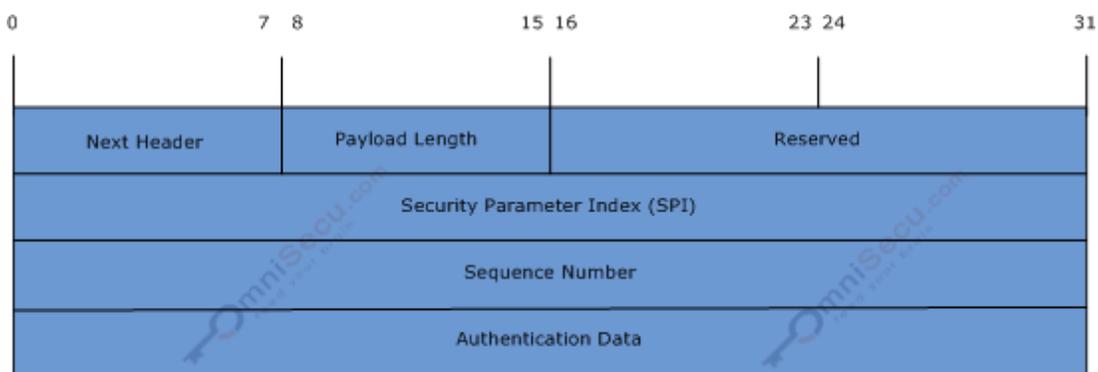


Figure: Authentication Header (AH) - Header

Next Header: Next header field points to next protocol header that follows the AH header. It can be a Encapsulating Security Payload (ESP) header, a TCP header or a UDP header (depending on the network application).

Payload Length: specifies the length of AH in 32-bit words (4-byte units), minus 2.

Reserved: This field is currently set to 0, reserved for future use.

Security Parameter Index (SPI): The Security Parameter Index (SPI) field contains the Security Parameter Index, is used to identify the security association used to authenticate this packet.

Sequence Number: Sequence Number field is the number of messages sent from the sender to the receiver using the current SA. The initial value of the counter is 1. The function of this field is to enable replay protection, if required.

Authentication Data: The Authentication Data field contains the result of the Integrity Check Value calculation, that can be used by the receiver to check the authentication and integrity of the packet. This field is padded to make total length of the AH is an exact number of 32-bit words. RFC 2402 requires that all AH implementations support at least HMAC-MD5-96 and HMAC-SHA1-96.

Combining Security Associations

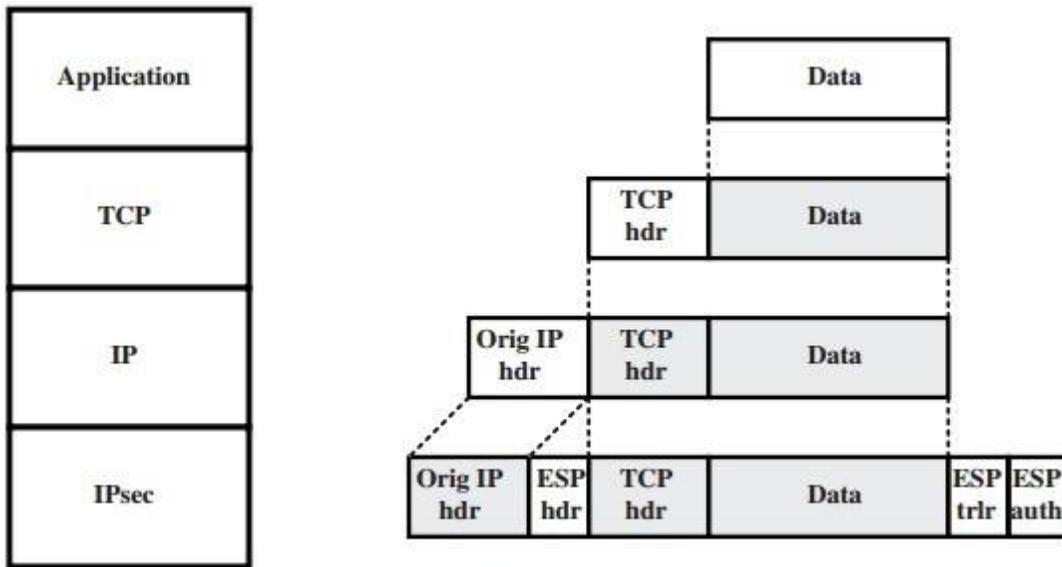
An individual SA can implement either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for the services provided by both AH and ESP. Further, a particular traffic flow may require IPsec services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPsec services. The term *security association bundle* refers to a sequence of SAs through which traffic must be processed to provide a desired set of IPsec services. The SAs in a bundle may terminate at different endpoints or at the same endpoints.

Security associations may be combined into bundles in two ways:

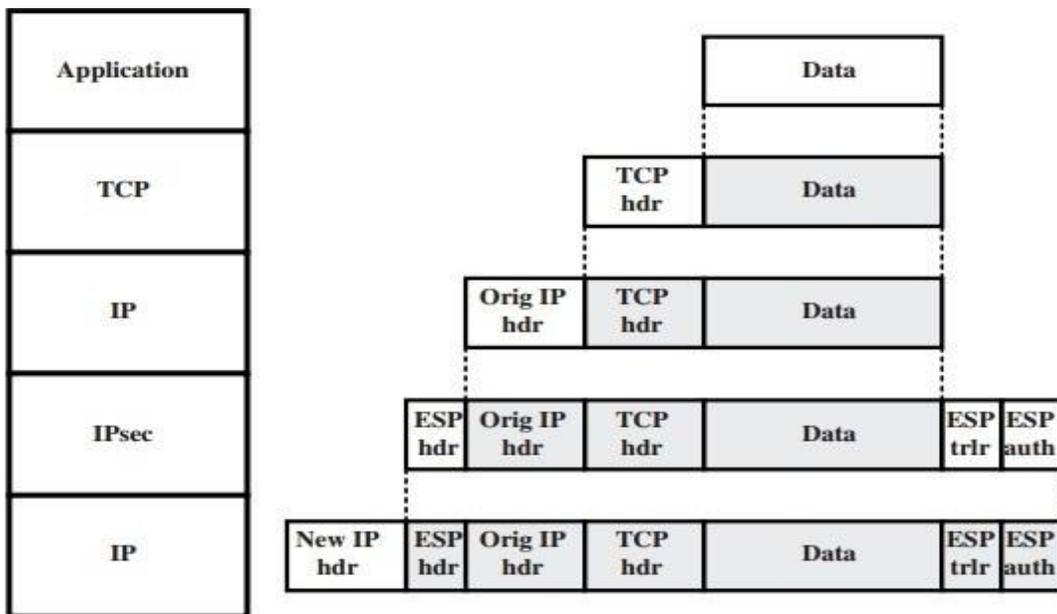
Transport adjacency: Refers to applying more than one security protocol to the same IP packet without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit since the processing is performed at one

IPsec instance: the (ultimate) destination.

Iterated tunneling: Refers to the application of multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPsec site along the path.



(a) Transport mode



(b) Tunnel mode

Figure 19.9 Protocol Operation for ESP

The two approaches can be combined, for example, by having a transport SA between hosts travel part of the way through a tunnel SA between security gateways. One interesting issue that arises when considering SA bundles is in the order in which authentication and encryption may be applied between a given pair of endpoints and the ways of doing so. We examine that issue next. Then we look at combinations of SAs that involve at least one tunnel.

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). These are illustrated in Figure 19.10.

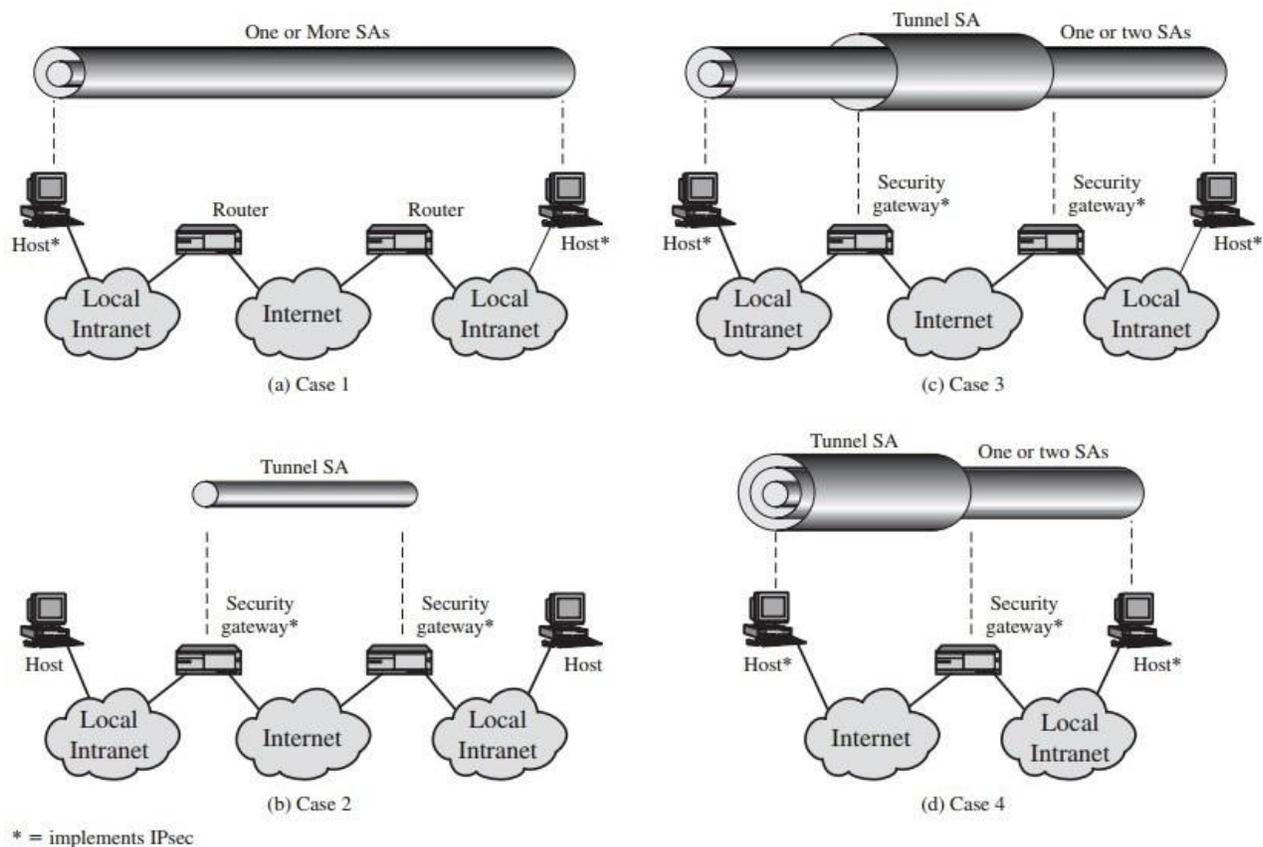


Figure 19.10 Basic Combinations of Security Associations

The lower part of each case in the figure represents the physical connectivity of the elements; the upper part represents logical connectivity via one or more nested SAs. Each

SA can be either AH or ESP. For host-to-host SAs, the mode may be either transport or tunnel; otherwise it must be tunnel mode.

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 are

- AH in transport mode
- ESP in transport mode
- ESP followed by AH in transport mode (an ESP SA inside an AH SA)
- Any one of a, b, or c inside an AH or ESP in tunnel mode

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. This builds on case 2 by adding end-to-end security. The same combinations discussed for cases 1 and 2 are allowed here. The gateway-to-gateway tunnel provides either authentication, confidentiality, or both for all traffic between end systems. When the gateway-to-gateway tunnel is ESP, it also provides a limited form of traffic confidentiality. Individual hosts can implement any additional IPsec services required for given applications or given users by means of end-to-end SAs.

Case 4. This provides support for a remote host that uses the Internet to reach an organization's firewall and then to gain access to some server or workstation behind the firewall. Only tunnel mode is required between the remote host and the firewall. As in case 1, one or two SAs may be used between the remote host and the local host.

IPSec Services

IPSec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. Two protocols are used to provide security: an authentication protocol designated by the header of the protocol, Authentication Header (AH); and a combined encryption/authentication protocol designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP).

The services are

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

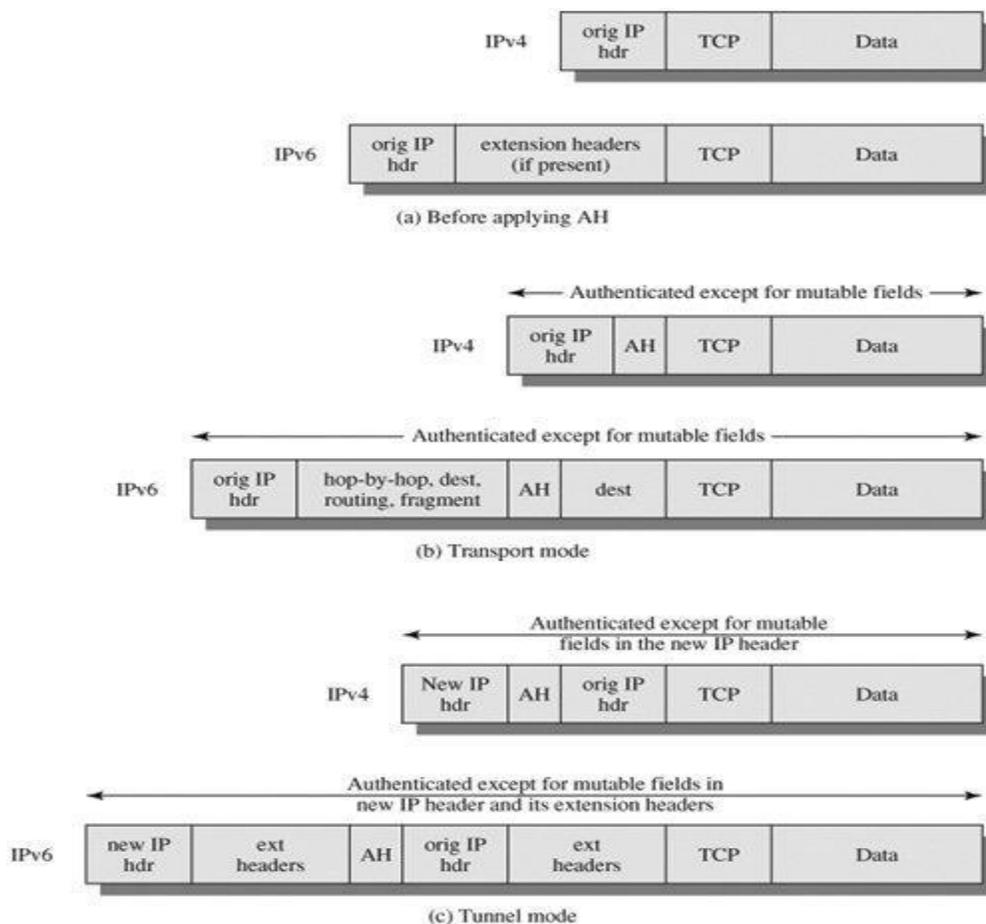
Transport and Tunnel Modes Transport Mode

Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet. Examples include a TCP or UDP segment or an ICMP packet, all of which operate directly above IP in a host protocol stack. Typically, transport mode is used for end-to-end communication between two hosts (e.g., a client and a server, or two workstations). When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, the payload is the data that normally follow both the IP header and any IPv6 extensions headers that are present, with the possible exception of the destination options header, which may be included in the protection. ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

Tunnel Mode

Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header. The entire original, or inner,

packet travels through a "tunnel" from one point of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security. Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewall or router that implements IPSec. With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPSec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPSec software in the firewall or secure router at the boundary of the local network.



Internet Key Exchange (IKE)

Internet Key Exchange (IKE) is a key management protocol standard used in conjunction with the Internet Protocol Security (IPSec) standard protocol. It provides security for virtual private networks' (VPNs) negotiations and network access to remote hosts. It can also be described as a method for exchanging keys for encryption and authentication over an unsecured medium, such as the Internet.

IKE is a hybrid protocol based on:

- ISAKMP (RFC2408): Internet Security Association and Key Management Protocols are used for negotiation and establishment of security associations. This protocol establishes a secure connection between two IPSec peers.
- Oakley (RFC2412): This protocol is used for key agreement or key exchange. Oakley defines the mechanism that is used for key exchange over an IKE session. The default algorithm for key exchange used by this protocol is the Diffie-Hellman algorithm.
- SKEME: This protocol is another version for key exchange.

IKE enhances IPsec by providing additional features along with flexibility. IPsec, however, can be configured without IKE.

IKE has many benefits. It eliminates the need to manually specify all the IPSec security parameters at both peers. It allows the user to specify a particular lifetime for the IPsec security association. Furthermore, encryption can be changed during IPsec sessions. Moreover, it permits certification authority. Finally, it allows dynamic authentication of peers.

UNIT –IV: WEB SECURITY

WEB SECURITY CONSIDERATIONS

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. As such, the security tools and approaches discussed so far in this book are relevant to the issue of Web security. But, as pointed out in the Web presents new challenges not generally appreciated in the context of computer and network security.

The Internet is two-way. Unlike traditional publishing environments—even electronic publishing systems involving teletext, voice response, or fax-back the Web is vulnerable to attacks on the Web servers over the Internet.

- The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.
- Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws. The short history of the Web is filled with examples of new and upgraded systems, properly installed, that are vulnerable to a variety of security attacks.
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.

- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

Web Security Threats

The following table provides a summary of the types of security threats faced when using the Web. One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site.

Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server. Issues of server and browser security fall into the category of computer system security; Part Four of this book addresses the issue of system security in general but is also applicable to Web system security.

	Threats	Consequences	Countermeasures
Integrity	<ul style="list-style-type: none"> • Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit 	<ul style="list-style-type: none"> • Loss of information • Compromise of machine • Vulnerability to all other threats 	Cryptographic checksums
Confidentiality	<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • Loss of information • Loss of privacy 	Encryption, Web proxies
Denial of Service	<ul style="list-style-type: none"> • Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks 	<ul style="list-style-type: none"> • Disruptive • Annoying • Prevent user from getting work done 	Difficult to prevent
Authentication	<ul style="list-style-type: none"> • Impersonation of legitimate users • Data forgery 	<ul style="list-style-type: none"> • Misrepresentation of user • Belief that false information is valid 	Cryptographic techniques

Table: Comparison of threats on the web

Web Traffic Security Approaches

A number of approaches to providing Web security are possible. The various approaches that have been considered are similar in the services they provide and, to some extent, in the mechanisms that they use, but they differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack.

The below figure illustrates this difference. One way to provide Web security is to use IP security (IPsec). The advantage of using IPsec is that it is transparent to end users and

applications and provides a general-purpose solution. Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

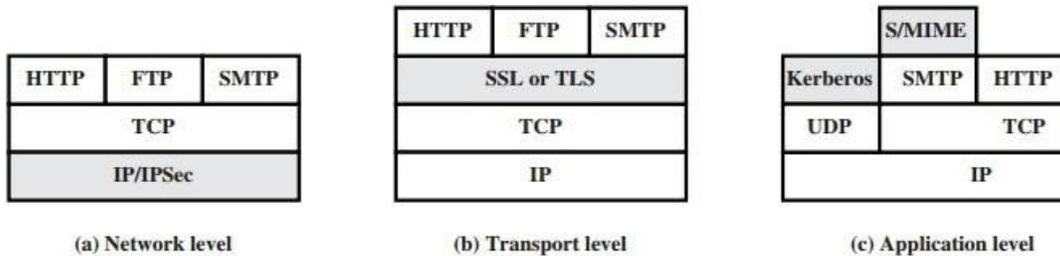


Figure: Relative location of security facilities in the TCP/IP protocol stack

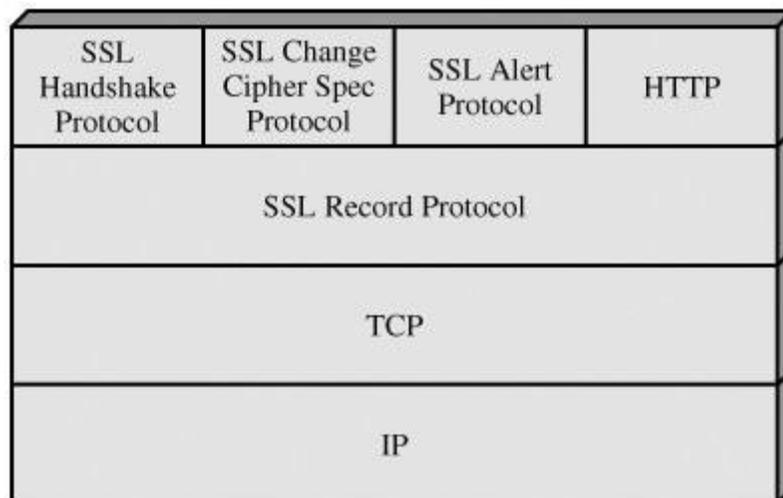
Areas:

1. SSL – Secure Socket Layer
2. TLS – Transport Layer Security
3. SET – Secure Electronic Transaction

SECURE SOCKET LAYER AND TRANSPORT LAYER SECURITY

SSL Architecture

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



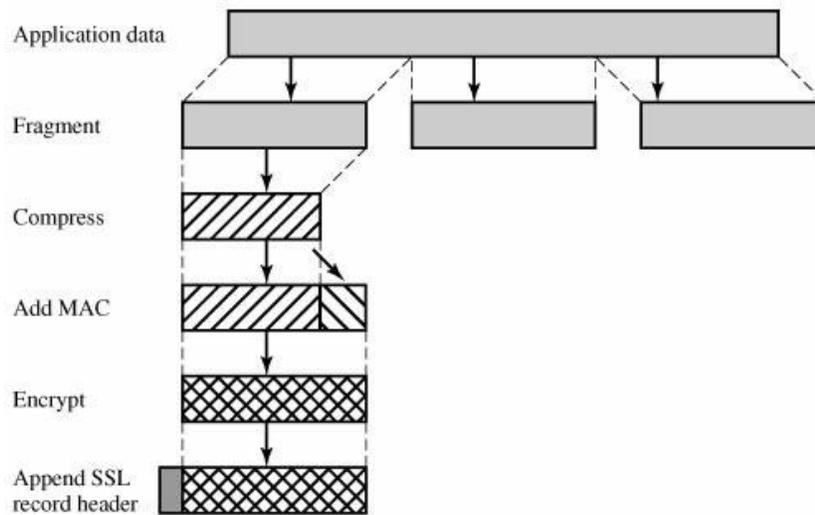
The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: 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.

SSL Record Protocol

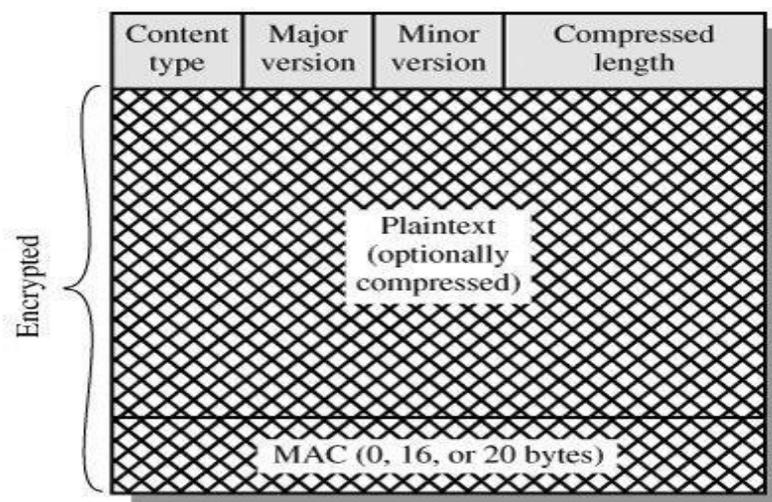
The SSL Record Protocol provides two services for SSL connections:

- **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 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. In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null. The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used.

SSL Record Format



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

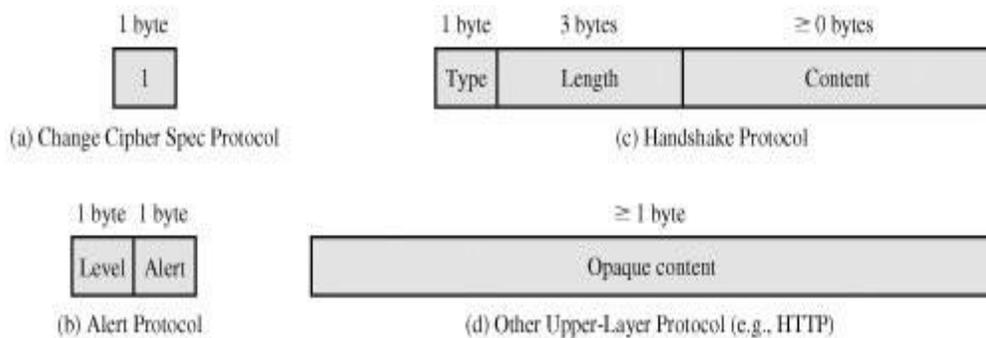
Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.

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 $2^{14} + 2048$. The content types that have been defined are change_cipher_spec, alert, handshake, and application_data.

SSL Record Protocol Payload 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.



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.

Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted.

Transport Layer Security (TLS)

Transport Layer Security (TLS) are designed to provide security at the transport layer. TLS was derived from a security protocol called Secure Service Layer (SSL). TLS ensures that no third party may eavesdrops or tamper with any message.

There are several benefits of TLS:

- Encryption: TLS/SSL can help to secure transmitted data using encryption.
- Interoperability: TLS/SSL works with most web browsers, including Microsoft Internet Explorer and on most operating systems and web servers.
- Algorithm flexibility: TLS/SSL provides operations for authentication mechanism, encryption algorithms and hashing algorithm that are used during the secure session.
- Ease of Deployment: Many applications TLS/SSL temporarily on a windows server 2003 operating systems.
- Ease of Use: Because we implement TLS/SSL beneath the application layer, most of its operations are completely invisible to client.

Working of TLS:

The client connect to server (using TCP), the client will be something. The client sends number of specification:

1. Version of SSL/TLS.
2. Which cipher suites, compression method it wants to use.

The server checks what the highest SSL/TLS version is that is supported by them both, picks a cipher suite from one of the clients option (if it supports one) and optionally picks a compression method. After this the basic setup is done, the server provides its certificate. This certificate must be trusted either by the client itself or a party that the client trusts. Having verified the certificate and being certain this server really is who he claims to be (and not a man in the middle), a key is exchanged. This can be a public key, “PreMasterSecret” or simply nothing depending upon cipher suite.

Both the server and client can now compute the key for symmetric encryption. The handshake is finished and the two hosts can communicate securely. To close a connection by finishing. TCP connection both sides will know the connection was improperly terminated. The connection cannot be compromised by this through, merely interrupted.

Difference between Secure Socket Layer (SSL) and Transport Layer Security (TLS)

SSL stands for Secure Socket Layer while TLS stands for Transport Layer Security. Both Secure Socket Layer and Transport Layer Security are the protocols used to provide the security between web browser and web server.

The main differences between Secure Socket Layer and Transport Layer Security are that. In SSL (Secure Socket Layer), Message digest is used to create master secret and It provides the basic security services which are **Authentication** and **confidentiality**, while In TLS (Transport Layer Security), Pseudo-random function is used to create master secret.

SECURE ELECTRONIC TRANSACTION

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. The current version, SETv1, emerged from a call for security standards by MasterCard and Visa in February 1996. A wide range of companies were involved in developing the initial specification, including IBM, Microsoft, Netscape, RSA, Terisa, and Verisign.

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 transaction when and where necessary

Key Features of SET

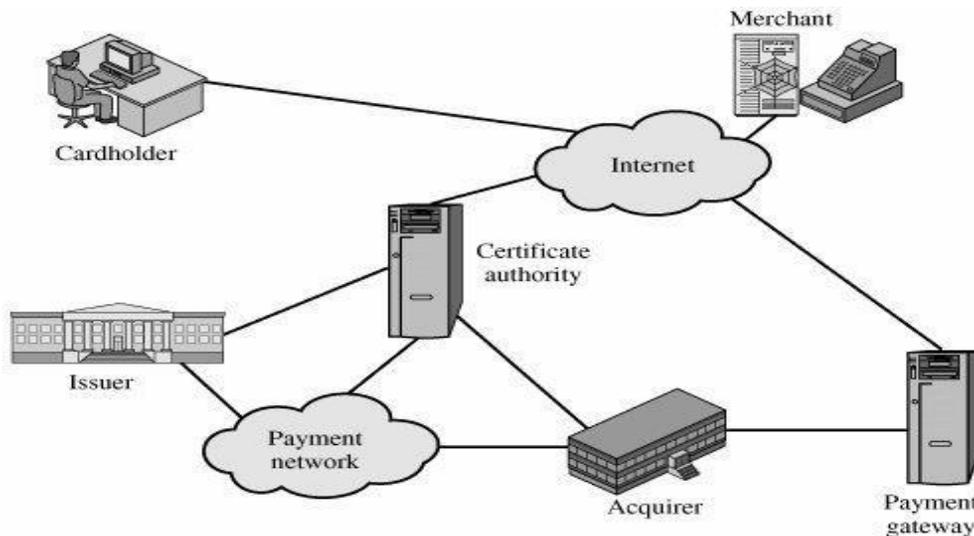
To meet the requirements just outlined, SET incorporates the following features:

- **Confidentiality of information:** Cardholder account and payment information is secured as it travels across the network. An interesting and important feature of SET is that it prevents the merchant from learning the cardholder's credit card number; this is only provided to the issuing bank. Conventional encryption by DES is used to provide confidentiality.
- **Integrity of data:** Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions. SET guarantees that these message contents are not altered in transit. RSA digital signatures, using

SHA-1 hash codes, provide message integrity. Certain messages are also protected by HMAC using SHA-1.

- **Cardholder account authentication:** SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.

- **Merchant authentication:** SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards. SET uses X.509v3 digital certificates with RSA signatures for this purpose.



SET Participants

- **Cardholder:** In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.

- **Merchant:** A merchant is a person or organization that has goods or services to sell to the cardholder. Typically, these goods and services are offered via a Web site or by electronic mail. A merchant that accepts payment cards must have a relationship with an acquirer.
- **Issuer:** This is a financial institution, such as a bank, that provides the cardholder with the payment card. Typically, accounts are applied for and opened by mail or in person.
- **Acquirer:** This is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments. Merchants will usually accept more than one credit card brand but do not want to deal with multiple bankcard associations or with multiple individual issuers. The acquirer provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit. The acquirer also provides electronic transfer of payments to the merchant's account. Subsequently, the acquirer is reimbursed by the issuer over some sort of payment network for electronic funds transfer.
- **Payment gateway:** This is a function operated by the acquirer or a designated third party that processes merchant payment messages. The payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system.
- **Certification authority (CA):** This is an entity that is trusted to issue

X.509v3 public-key certificates for cardholders, merchants, and payment gateways. The success of SET will depend on the existence of a CA infrastructure available for this purpose. A hierarchy of CAs is used, so that participants need not be directly certified by a root authority.

The sequence of events those are required for a transaction

- 1. The customer opens an account.** The customer obtains a credit card account, such as MasterCard or Visa, with a bank that supports electronic payment and SET.
- 2. The customer receives a certificate.** After suitable verification of identity, the customer receives an X.509v3 digital certificate, which is signed by the bank. The certificate verifies the customer's RSA public key and its expiration date. It also establishes a relationship, guaranteed by the bank, between the customer's key pair and his or her credit card.
- 3. Merchants have their own certificates.** A merchant who accepts a certain brand of card must be in possession of two certificates for two public keys owned by the merchant: one for signing messages, and one for key exchange. The merchant also needs a copy of the payment gateway's public-key certificate.
- 4. The customer places an order.** This is a process that may involve the customer first browsing through the merchant's Web site to select items and determine the price. The customer then sends a list of the items to be purchased to the merchant, who returns an order form containing the list of items, their price, a total price, and an order number.
- 5. The merchant is verified.** In addition to the order form, the merchant sends a copy of its certificate, so that the customer can verify that he or she is dealing with a

valid store.

6. The order and payment are sent. The customer sends both order and payment information to the merchant, along with the customer's certificate. The order confirms the purchase of the items in the order form. The payment contains credit card details. The payment information is encrypted in such a way that it cannot be read by the merchant. The customer's certificate enables the merchant to verify the customer.

7. The merchant requests payment authorization. The merchant sends the payment information to the payment gateway, requesting authorization that the customer's available credit is sufficient for this purchase.

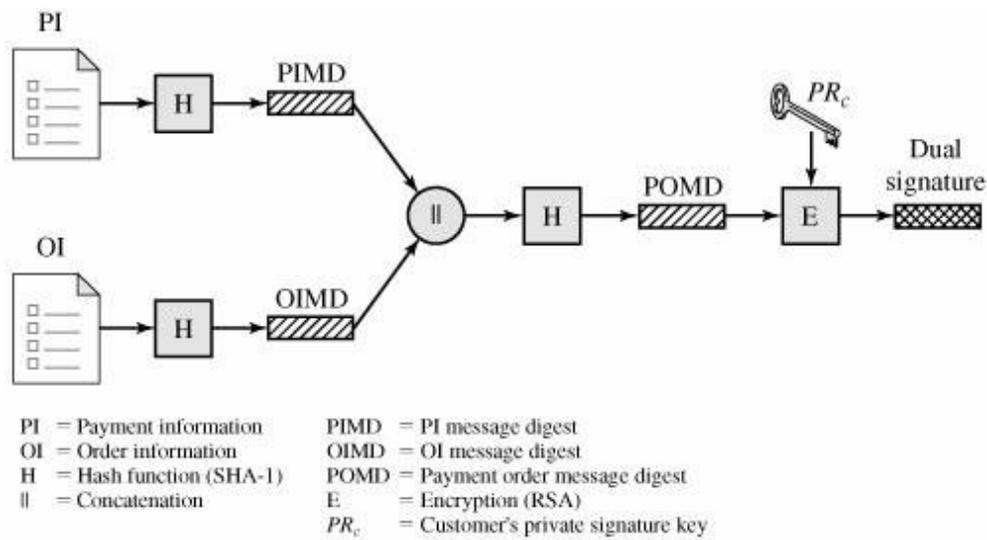
8. The merchant confirms the order. The merchant sends confirmation of the order to the customer.

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

10. The merchant requests payment. This request is sent to the payment gateway, which handles all of the payment processing.

Dual Signature

The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, 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.



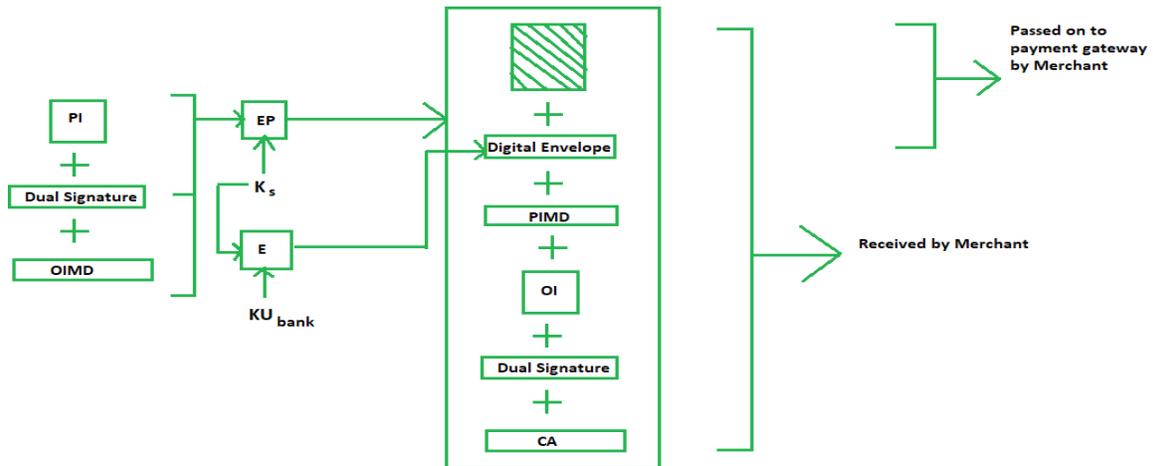
The customer is afforded extra protection in terms of privacy by keeping these two items separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. The link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service. To see the need for the link, suppose that the customers send the merchant two messages: a signed OI and a signed PI, and the merchant passes the PI on to the bank. If the merchant can capture another OI from this customer, the merchant could claim that this OI goes with the PI rather than the original OI. The linkage prevents this. 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.

Purchase Request Generation:

The process of purchase request generation requires three inputs:

- Payment Information (PI)
- Dual Signature
- Order Information Message Digest (OIMD)

The purchase request is generated as follows:



Here,

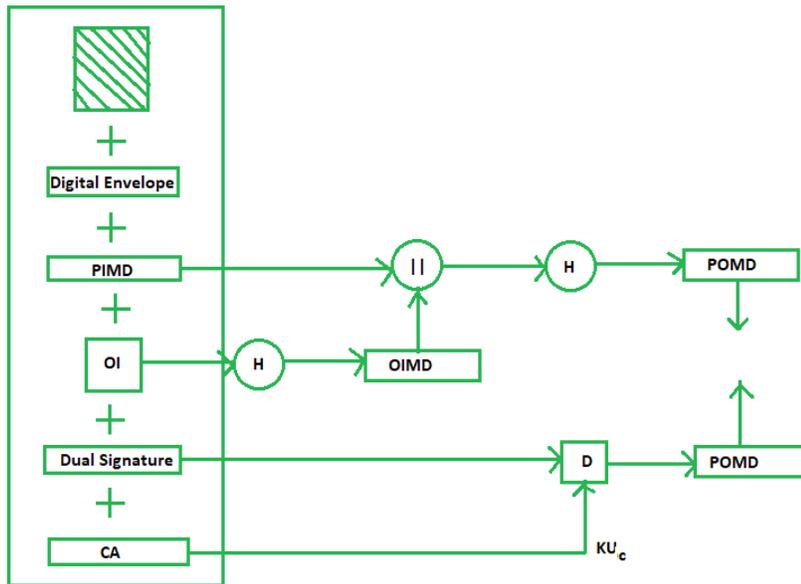
- PI, OIMD, OI all have the same meanings as before.

The new things are :

- EP which is symmetric key encryption
- Ks is a temporary symmetric key
- KUBank is public key of bank
- CA is Cardholder or customer Certificate
- Digital Envelope = $E(KUBank, Ks)$

Purchase Request Validation on Merchant Side:

The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows:



Since we used Customer private key in encryption here we use KU_c which is public key of customer or cardholder for decryption 'D'.

Payment Authorization and Payment Capture:

Payment authorization as the name suggests is the authorization of payment information by merchant which ensures payment will be received by merchant. Payment capture is the process by which merchant receives payment which includes again generating some request blocks to gateway and payment gateway in turn issues payment to merchant.

Basic concepts of SNMP

Network Management Architecture:

A network management system is a collection of tools for network monitoring and control that is integrated in the following senses:

- A single operator interface with a powerful but user-friendly set of commands for performing most or all network management tasks. .
 - A minimal amount of separate equipment. That is, most of the hardware and software required for network management is incorporated into the existing user equipment.
- ⇒ A network management system consists of incremental hardware and software additions implemented among existing network components.
- ⇒ The software used in accomplishing the network management tasks resides in the host computers and communications processors.
- ⇒ A network management system is designed to view the entire network as a unified architecture, with addresses and labels assigned to each point and the specific attributes of each element and link known to the system.

The model of network management that is used for SNMP includes the following key elements:

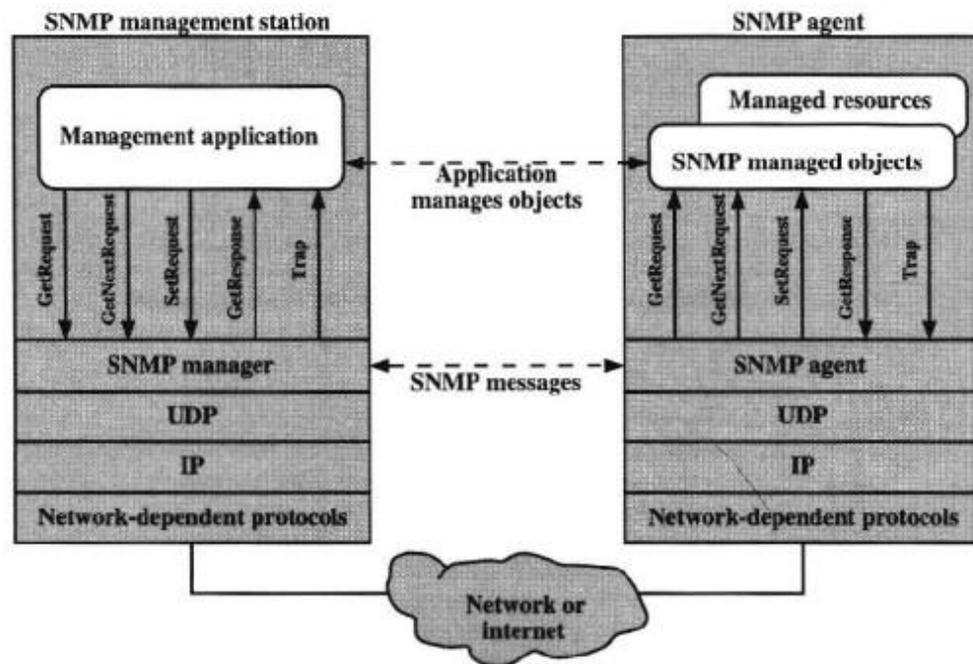
- Management station
 - Management agent
 - Management information base
 - Network management protocol
- ⇒ The **management station** is typically a stand-alone device that serves as the interface for the human network manager into the network management system.
- ⇒ The **management agent** responds to requests for information from a management station, responds to requests for actions from the management station, and may asynchronously provide the management station with important but unsolicited information.
- ⇒ To manage resources in the network, each resource is represented as an object. An object is, essentially, a data variable that represents one aspect of the managed agent. The collection of objects is referred to as a **management information base (MIB)**.
- ⇒ The management station and agents are linked by a **network management protocol**. The protocol used for the management of TCP/IP networks is the Simple Network Management Protocol (SNMP). This protocol includes the following key capabilities:

- **Get:** Enables the management station to retrieve the value of objects at the agent
- **Set:** Enables the management station to set the value of objects at the agent
- **Notify:** Enables an agent to notify the management station of significant events

Network Management Protocol Architecture

SNMP is a simple tool for network management. It defines a limited, easily implemented management information base (MIB) of scalar variables and two-dimensional tables, and it defines a streamlined protocol to enable a manager to get and set MIB variables and to enable an agent to issue unsolicited notifications, called *traps*.

SNMP was designed to be an application-level protocol that is part of the TCP/IP protocol suite. It is intended to operate over the User Datagram Protocol (UDP), defined in RFC 768.



7.2 SNMPv1 COMMUNITY FACILITY

SNMP network management has several characteristics not typical of all distributed applications. The application involves a one-to-many relationship between a manager and a set of agents: The manager is able to get and set objects in the agents and is able to receive traps from the agents. Thus, from an operational or control point of view, the manager "manages" a number of agents. There may be a number of managers, each of which manages all or a subset of the agents in the configuration. These subsets may overlap.

Each agent controls its own local MIB, and must be able to control the use of that MIB by a number of managers.

There are three aspects of this control:

- **Authentication service:** The agent may wish to limit access to the MIB to authorized managers.
 - **Access policy:** The agent may wish to give different access privileges to different managers.
 - **Proxy service:** An agent may act as proxy to other agents. This may involve implementing the authentication service and/or access policy for the other agents on the proxy system.
- An **SNMP community** is a relationship between an SNMP agent and a set of SNMP managers that defines authentication, access control, and proxy characteristics.
 - The community concept is a local one, defined at the agent.
 - The agent establishes one community for each desired combination of authentication, access control, and proxy characteristics.
 - Each community is given a unique (within this agent) community name, and the managers within that community are provided with and must employ the community name in all get and set operations.
 - The agent may establish a number of communities, with overlapping manager membership.

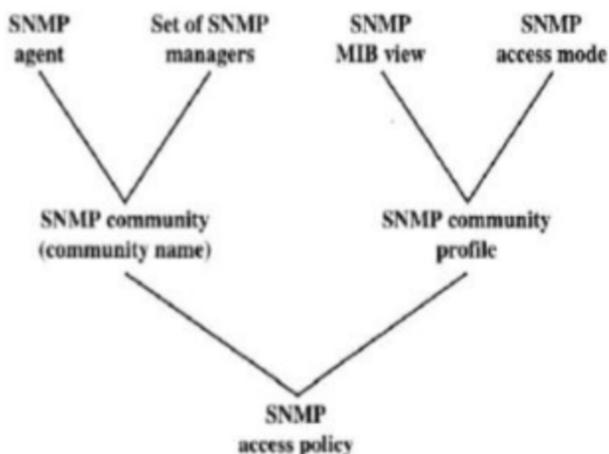
Authentication Service

The purpose of the SNMPv1 authentication service is to assure the recipient that an SNMPv1 message is from the source that it claims to be from. SNMPv1 only provides for a trivial scheme for authentication. Every message (get or put request) from a manager to an agent includes a community name. This name functions as a password, and the message is assumed to be authentic if the sender knows the password.

Access Policy

By defining 'a community, an agent limits access to its MIB to a selected set of managers. By the use of more than one community, the agent can provide different categories of MIB access to different managers. There are two aspects to this access control:

- **SNMP MIB view:** A subset of the objects within an MIB. Different MIB views may be defined for each community. The set of objects in a view need not belong to a single sub-tree of the MIB.
- **SNMP access mode:** An element of the set {READ-ONLY, READ-WRITE}. An access mode is defined for each community.

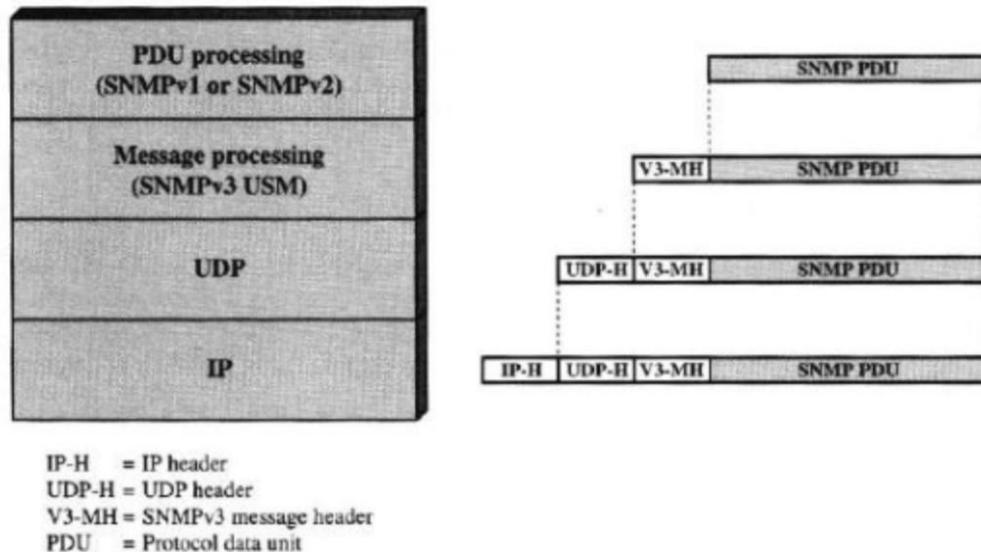


SNMPv1 Administrative Concepts

SNMPv3

SNMPv3 defines a security capability to be used in conjunction with SNMPv2 (preferred) or SNMPv1.

Fig indicates the relationship among the different versions of SNMP by means of the formats involved.



SNMP Architecture:

SNMP architecture defined in standard RFC 2571 consists of SNMP entities. These entities are interactive and are organized as an abstract set of functions and parameters that are used for passing control and data information. They act either as an agent node, manager node or both. SNMP entities comprises of collection of individual units that communicate with each other in order to provide functions.

The RFC 2571 architecture reflects a key design requirement for SNMPv3:

Design a modular architecture that

1. It allows minimum and cheaper services to be implemented over broad spectrum of functioning surrounding.
2. It is possible to move some part of the architecture forward in a conventional way even though general agreements have not reached all its part.
3. It is possible to adopt other security models.

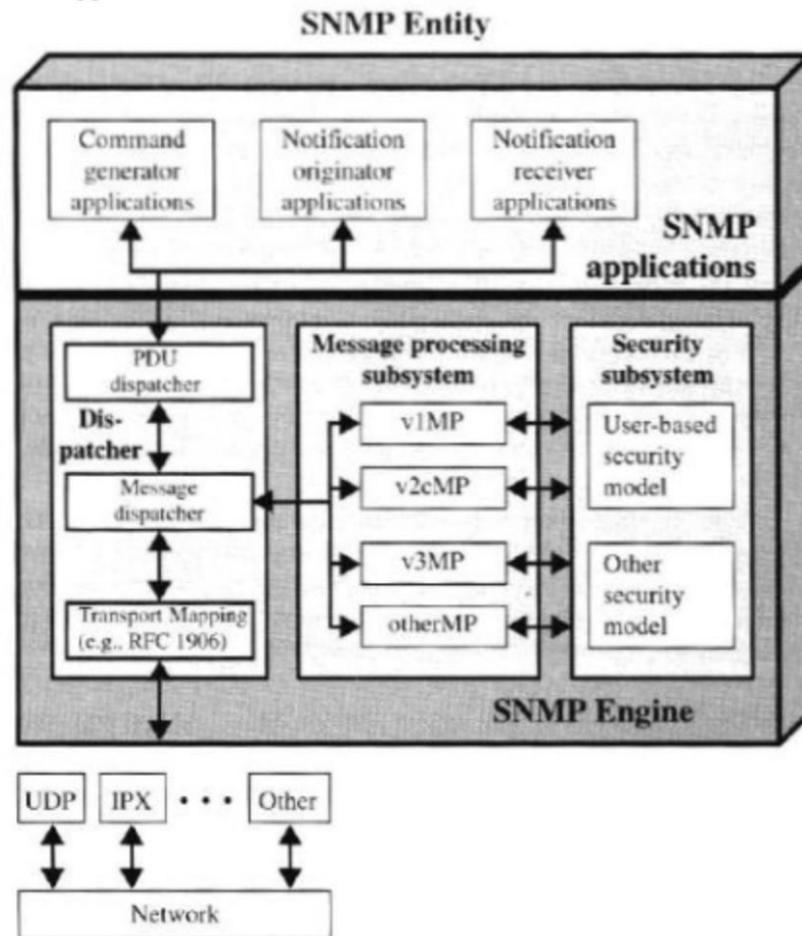
❖ SNMP Entity

Each SNMP entity includes a single SNMP engine. An SNMP engine implements functions for sending and receiving messages, authenticating and encrypting/decrypting messages, and controlling access to managed objects. These functions are provided as services to one or more applications that are configured with the SNMP engine to form an SNMP entity.

❖ **Traditional SNMP manager;**

A traditional SNMP manager interacts with SNMP agents by issuing commands (get, set) and by receiving trap messages; the manager may also interact with other managers by issuing Inform Request PDUs, which provide alerts, and by receiving Inform Response PDUs, which acknowledge Inform Requests. In SNMPv3 terminology, a traditional SNMP manager includes three categories of applications:

1. Command Generator Application
2. Notification Originator Application
3. Notification Receiver Application



Command Generator Application: This application examines and modifies the management data of remote agents. It utilizes SNMPv1 and/or SNMPv2 processing module containing Get, GetBulk, GetNext and SetMessages.

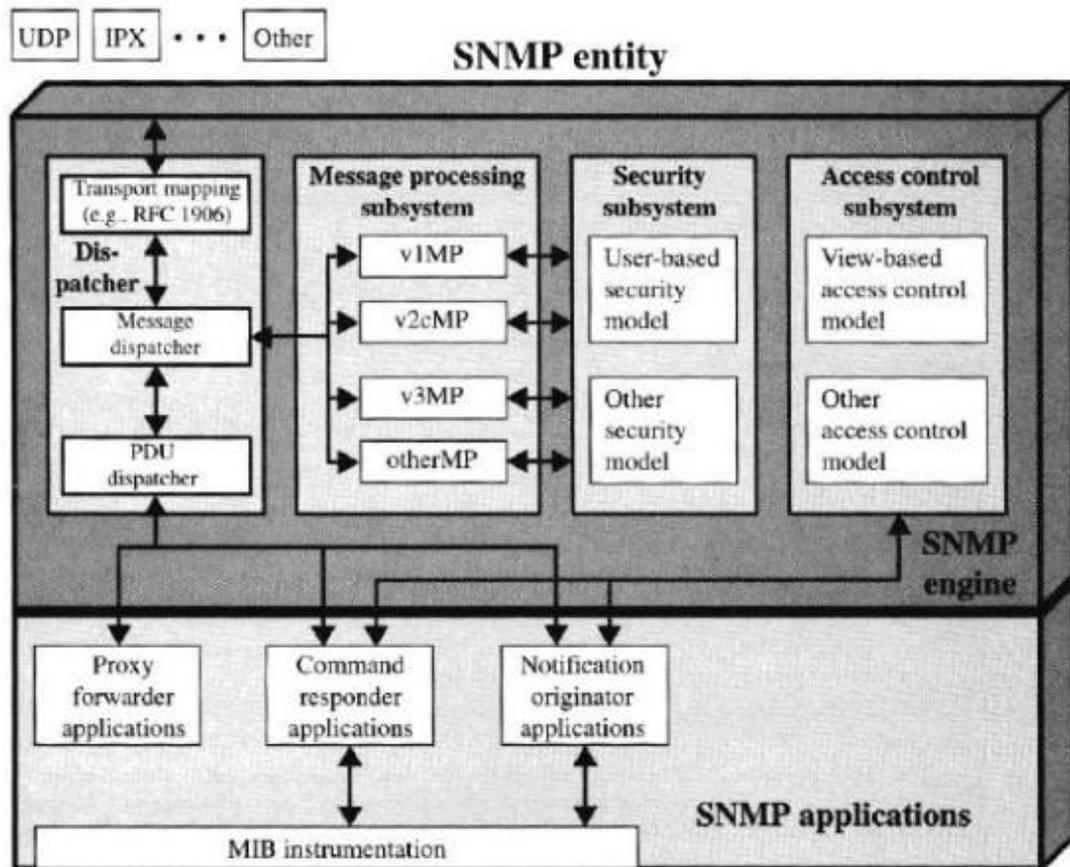
Notification Originator Application: In case of traditional SNMP manager, this application is responsible for starting the transmission of asynchronous messages like InformRequest PDU.

Notification Receiver Application: It is responsible for processing incoming asynchronous messages which may be either InformRequest PDU, SNMPv1 Trap PDU's or SNMPv2 Trap PDU.

❖ **Traditional SNMP Agent:**

SNMP agent consists of three kinds of applications. They are as follows:

1. Command Responder Application
2. Notification Originator Application
3. Proxy Forwarder Application



Command Responder Application: This application make provisions for accessing management data. It is responsible for responding to every incoming request PDU. It does this by restoring the managed entity and/or by defining the managed entity.

Notification Originator Application: In case of traditional SNMP agent, this application is used for starting the transmission of asynchronous message like, Trap PDU of both SNMPv1, SNMPv2.

Proxy Forwarder Application: This application is responsible for forwarding messages between the entities.

User Security Model

- **Authentication:** Provides data integrity and data origin authentication. The message authentication code HMAC, with either the hash function MD5 or SHA1 provides authentication.
- **Timeliness:** Protects against message delay or replay.
- **Privacy:** Protects against disclosure of message payload. The cipher block chaining (CBC) mode of DES is used for encryption.
- **Message format:** Defines format of msg Security Parameters field, which supports the functions of authentication, timeliness, and privacy.
- **Discovery:** Defines procedures by which one SNMP engine obtains information about another SNMP engine.
- **Key management:** Defines procedures for key generation, update, and use.

USM Security Parameters:

Usm Security Parameters that specifies the internal format of the msg Security Parameters field in SNMPv3 message.

Authoritative SNMP Engine:

In any message transmission, one of the two entities, transmitter or receiver; is designated as the authoritative SNMP engine, according to the following rules.

- When an SNMP message contains a payload which expects a response, then the receiver of such messages is authoritative.

- When an SNMP message contains a payload which does not expect a response (for example an SNMPv2-Trap, Response, or Report PDU), then the sender of such a message is authoritative.

Thus, for messages sent on behalf of a Command Generator and for Inform messages from a Notification Originator, the receiver is authoritative.

This designation serves two purposes. The first one is,

- The timeliness of a message is determined with respect to a clock maintained by the authoritative engine.
- When an authoritative engine sends a message (Trap, Response, Report), it contains the current value of its clock, so that the non authoritative recipient can synchronize on that clock.
- When a non authoritative engine sends a message (Get, GetNext, GetBulk, Set, Inform), it includes its current estimate of the time value at the destination, allowing the destination to assess the message's timeliness.

The second one is,

- A key localization process, described later, enables a single principal to own keys stored in multiple engines; these keys are localized to the authoritative engine in such a way that the principal is responsible for a single key but avoids the security risk of storing multiple copies of the same key in a distributed network.

- It makes sense to designate the receiver of Command Generator and Inform PDUs as the authoritative engine, and therefore the possessor of the authoritative clock in an exchange.
- If a response or trap is delayed or replayed, little harm should occur. However, Command Generator and, to some extent, Inform PDUs result in management operations, such as reading or setting MIB objects.
- Thus, it is important to guarantee that such PDUs are not delayed or replayed, which could cause undesired effects

Elements of Usm Security Parameters :

When an outgoing message is passed to the USM by the Message Processor, the USM fills in the msgSecurity Parameters field. When an incoming message is passed to the USM by the Message Processor, the USM processes the values contained in msgSecurity Parameters.

The security parameters field consists of the following elements:

Msg Authoritative Engine ID: The snmp Engine ID of the authoritative SNMP engine involved in the exchange of this message. Thus, this value refers to the source for a Trap, Response, or Report, and to the destination for a Get, GetNext, GetBulk, Set, or Inform.

Msg Authoritative Engine Boots: The snmp Engine Boots value of the authoritative SNMP engine involved in the exchange of this message. The object snmp Engine Boots is an integer in the range 0 through $(2)^{31}-1$ that represents the number of times that this SNMP engine has initialized or reinitialized itself since its initial configuration.

Msg Authoritative Engine Time: The snmp Engine Time value of the authoritative SNMP engine involved in the exchange of this message. The object snmp Engine Time is an integer in the range 0 through $(2)^{31} - 1$ that represents the number of seconds since this authoritative SNMP engine last incremented the snmp Engine Boots object. Each authoritative SNMP engine is responsible for incrementing its own snm Engine Time value once per second. A non authoritative engine is responsible for incrementing its notion of snmp Engine Time for each remote authoritative engine with which it communicates.

Msg User Name: The user (principal) on whose behalf the message is being exchanged.

Msg Authentication Parameters: Null if authentication is not being used for this exchange. Otherwise, this is an authentication parameter. For the current definition of USM, the authentication parameter is an HMAC message authentication code.

Msg Privacy Parameters: Null if privacy is not being used for this exchange. Otherwise, this is a privacy parameter. For the current definition of USM, the privacy parameter is a value used to form the value (IV) in the DES CBC algorithm.

UNIT –V: SYSTEM SECURITY

INTRUDERS

One of the most publicized attacks to security is the intruder, generally referred to as hacker or cracker. Three classes of intruders are as follows:

- **Masquerader** – an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.
- **Misfeasor** – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.
- **Clandestine user** – an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However there is no way in advance to know whether an intruder will be benign or malign.

An analysis of previous attack revealed that there were two levels of hackers:

- The high levels were sophisticated users with a thorough knowledge of the technology.
- The low levels were the „foot soldiers“ who merely use the supplied cracking programs with little understanding of how they work.

one of the results of the growing awareness of the intruder problem has been the establishment of a number of Computer Emergency Response Teams (CERT). these co-operative ventures collect information about system vulnerabilities and disseminate it to systems managers. Unfortunately, hackers can also gain access to CERT reports.

In addition to running password cracking programs, the intruders attempted to modify login software to enable them to capture passwords of users logging onto the systems.

Intrusion techniques

The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruders to acquire information that should be protected. In most cases, the information is in the form of a user password.

Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. The password files can be protected in one of the two ways:

- **One way encryption** – the system stores only an encrypted form of user's password. In practice, the system usually performs a one way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.
- **Access control** – access to the password file is limited to one or a very few accounts.

The following techniques are used for learning passwords.

- Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
- Exhaustively try all short passwords.
- Try words in the system's online dictionary or a list of likely passwords.
- Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.
- Try user's phone number, social security numbers and room numbers.
- Try all legitimate license plate numbers.
- Use a torjan horse to bypass restriction on access.
- Tap the line between a remote user and the host system.

Two principle countermeasures:

Detection – concerned with learning of an attack, either before or after its success. ☹

Prevention – challenging security goal and an uphill battle at all times.

INTRUSION DETECTION:

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
- An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Figure 5.2.1 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

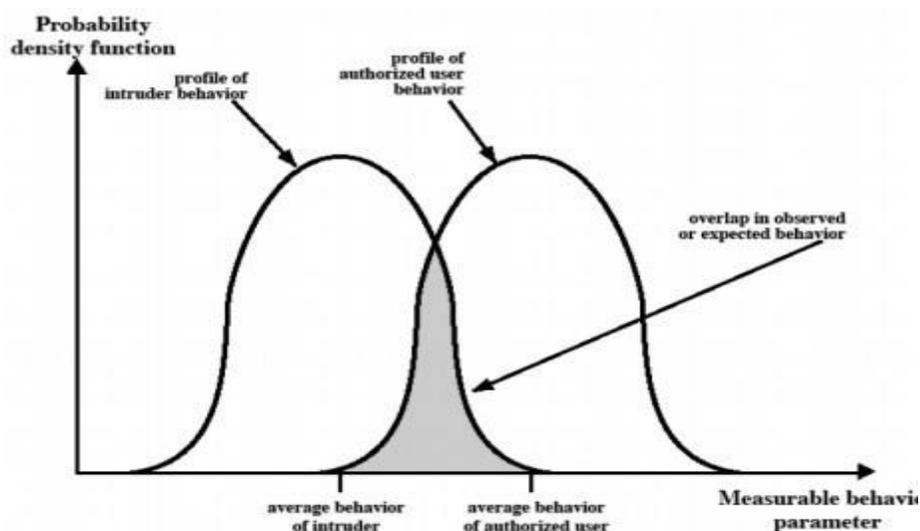


Fig. 5.2.1 Profiles of behavior of intruders and authorized users

1. The approaches to intrusion detection:

Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.

Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

Anomaly detection: Rules are developed to detect deviation from previous usage patterns.

Penetration identification: An expert system approach that searches for suspicious behavior.

In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders. On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, rule-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks.

Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

- **Native audit records:** Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.
- **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion

detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Each audit record contains the following fields:

- **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a
 - o process acting on behalf of users or groups of users.
- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures
- **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read
 - o or written, processor time, I/O units used, session elapsed time).
- **Time-Stamp:** Unique time-and-date stamp identifying when the action took place. Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command

COPY GAME.EXE TO <Library>GAME.EXE

issued by Smith to copy an executable file GAME from the current directory to the <Library> directory. The following audit records may be generated:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679
Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

In this case, the copy is aborted because Smith does not have write permission to <Library>. The decomposition of a user operation into elementary actions has three advantages:

Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus, the system can detect attempted subversions of access

Single-object, single-action audit records simplify the model and the implementation.

Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detection-specific audit records.

Statistical Anomaly Detection:

As was mentioned, statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems. **Threshold detection involves** counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.

Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined.

Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. Examples of metrics that are useful for profile-based intrusion detection are the following:

- **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.

- **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.

- **Interval timer:** The length of time between two related events. An example is the length of time between successive logins to an account.
- **Resource utilization:** Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution.

Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits.

- Mean and standard deviation
- Multivariate
- Markov process
- Time series
- Operational

The simplest statistical test is to measure the mean and standard deviation of a parameter over some historical period. This gives a reflection of the average behavior and its variability.

A multivariate model is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time).

A Markov process model is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.

A time series model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing.

Finally, an operational model is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits.

Rule-Based Intrusion Detection

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior.

As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past

Rule-based penetration identification takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.

Example heuristics are the following:

- o Users should not read files in other users' personal directories.
- o Users must not write other users' files.
- o Users who log in after hours often access the same files they used earlier.
- o Users do not generally open disk devices directly but rely on higher-level operating system utilities.
- o Users should not be logged in more than once to the same system.
- o Users do not make copies of system programs.

1 The Base-Rate Fallacy

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms.

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if

the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating.

2 Distributed Intrusion Detection

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN. Porras points out the following major issues in the design of a distributed intrusion detection system

A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.

One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data.

Either a centralized or decentralized architecture can be used.

Below figure shows the overall architecture, which consists of three main components:

- **Host agent module:** An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.
-
- **LAN monitor agent module:** Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
-
- **Central manager module:** Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

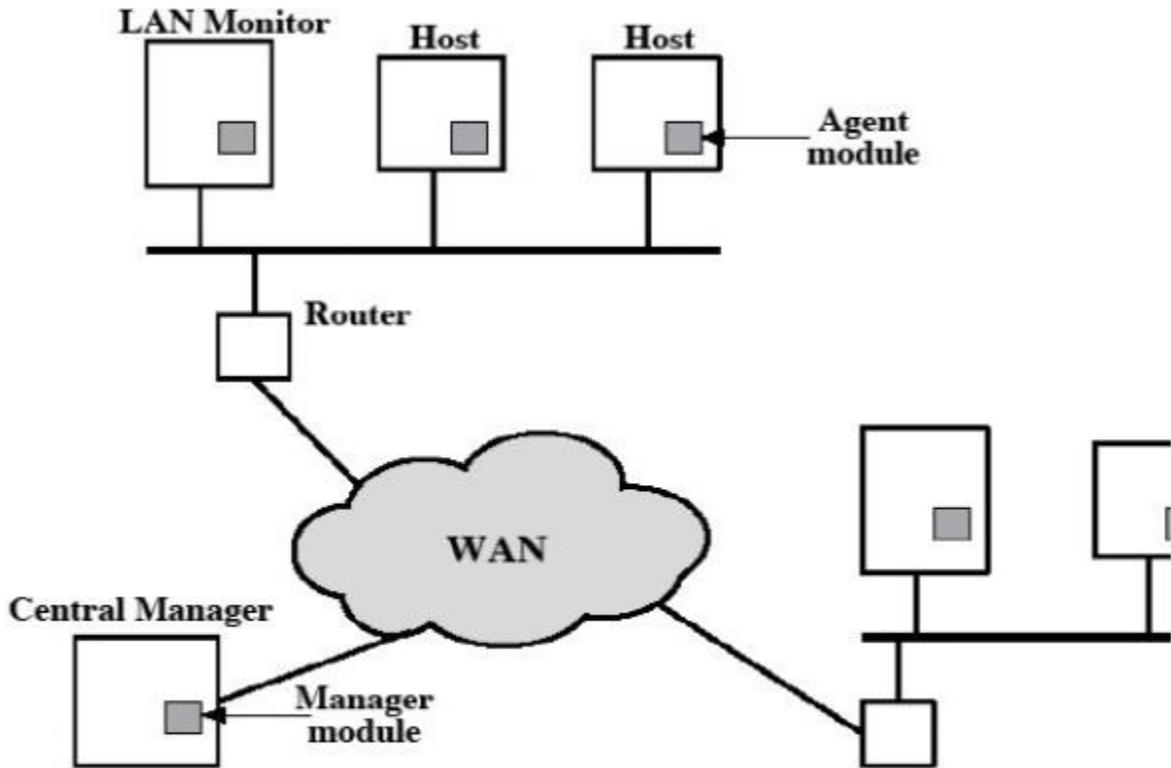


Fig. 5.2.3.1 Architecture of Distributed intrusion detection

The scheme is designed to be independent of any operating system or system auditing implementation.

- The agent captures each audit record produced by the native audit collection system.
- A filter is applied that retains only those records that are of security interest.
- These records are then reformatted into a standardized format referred to as the host audit record (HAR).
- Next, a template-driven logic module analyzes the records for suspicious activity.
- At the lowest level, the agent scans for notable events that are of interest independent of any past events.
- Examples include failed file accesses, accessing system files, and changing a file's access control.
- At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures).

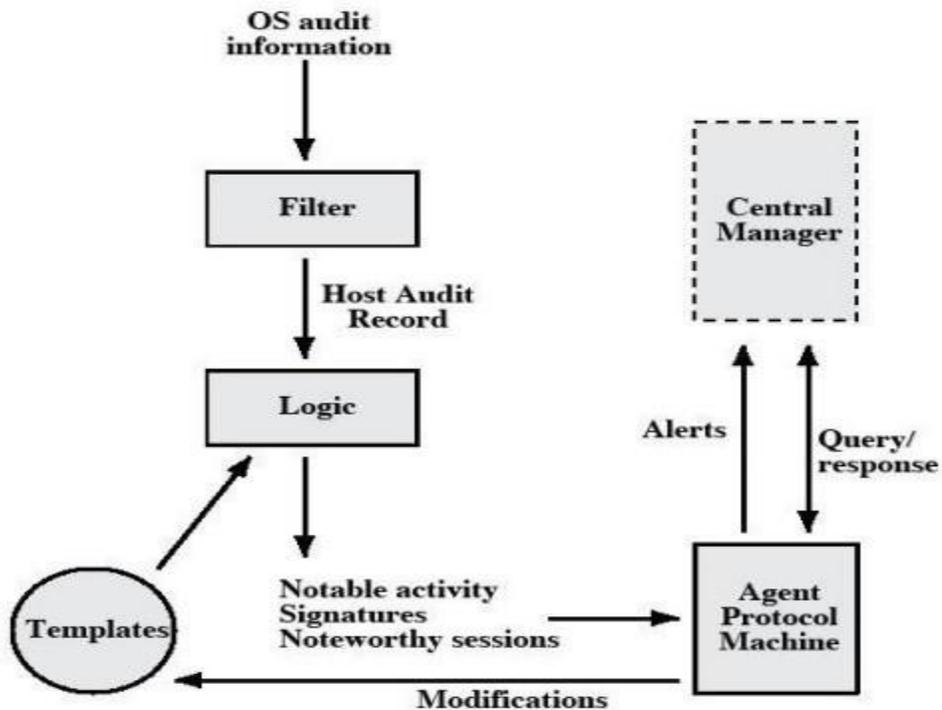
- Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.
- When suspicious activity is detected, an alert is sent to the central manager.
- The central manager includes an expert system that can draw inferences from received data.
- The manager may also query individual systems for copies of HARs to correlate with those from other agents.
- The LAN monitor agent also supplies information to the central manager.
- The LAN monitor agent audits host-host connections, services used, and volume of traffic.
- It searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as rlogin.

The architecture is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected.

3 Honeypots

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to

- divert an attacker from accessing critical systems
- collect information about the attacker's activity
- encourage the attacker to stay on the system long enough for administrators to respond



These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect.

4 Intrusion Detection Exchange Format

To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group.

The outputs of this working group include the following:

- a. A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements.
- b. A common intrusion language specification, which describes data formats that satisfy the requirements.
- c. A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised data formats relate to them.

PASSWORD MANAGEMENT

1. Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

- The ID determines whether the user is authorized to gain access to a system.
- The ID determines the privileges accorded to the user.
- The ID is used in ,what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

2. The Vulnerability of Passwords

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, the following procedure is employed.

- Each user selects a password of up to eight printable characters in length.
- This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine.
- The encryption routine, known as crypt(3), is based on DES. The DES algorithm is modified using a 12-bit "salt" value.
- Typically, this value is related to the time at which the password is assigned to the user.
- The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros.
- The output of the algorithm then serves as input for a second encryption.
- This process is repeated for a total of 25 encryptions.
- The resulting 64-bit output is then translated into an 11-character sequence.
- The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.

This method has been shown to be secure against a variety of cryptanalytic attacks

The salt serves three purposes:

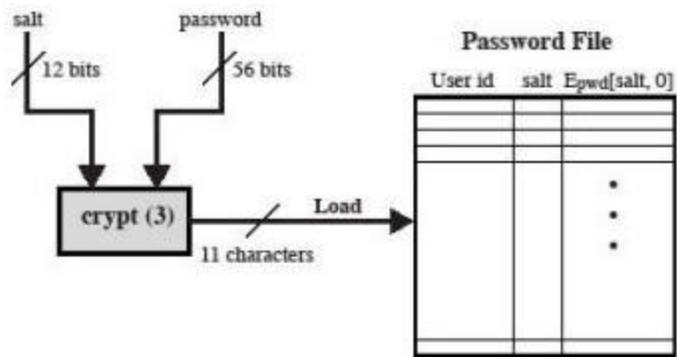
· It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.

· It effectively increases the length of the password without requiring the user to remember two additional characters.

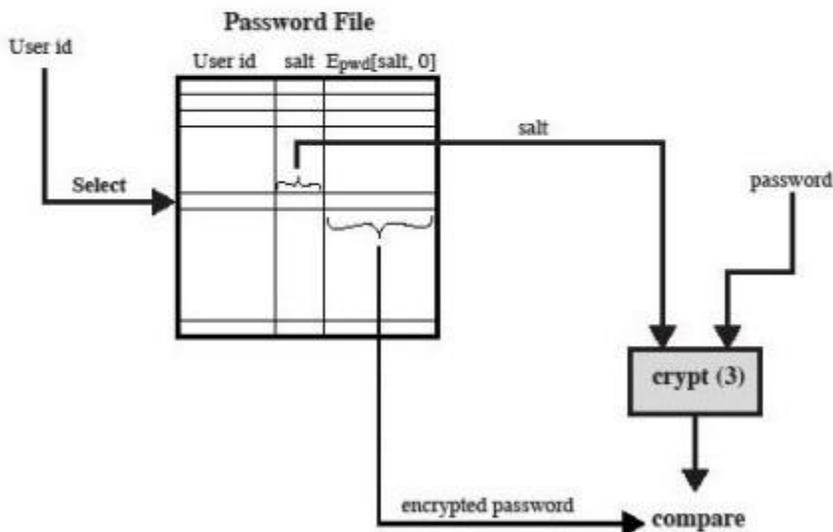
· It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.

When a user attempts to log on to a UNIX system, the user provides an ID and a password. The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password. The salt and user-supplied password are used as input to the encryption routine. If the result matches the stored value, the password is accepted. The encryption routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25.

Thus, there are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine.



(a) Loading a new password



(b) Verifying a password

Fig. 5.3.2.1 UNIX Password Scheme

As an example, a **password cracker was reported on the Internet in August 1993**. Using a Thinking Machines Corporation parallel computer, a performance of 1560 encryptions per second per vector unit was achieved. With four vector units per processing node (a standard configuration), this works out to 800,000 encryptions per second on a 128-node machine (which is a modest size) and 6.4 million encryptions per second on a 1024-node machine.

Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward.

Following strategy was used:

- Try the user's name, initials, account name, and other relevant personal information. In all, 130 different permutations for each user were tried.
- Try words from various dictionaries.
- Try various permutations on the words from step 2.
- Try various capitalization permutations on the words from step 2 that were not considered in step 3. This added almost 2 million additional words to the list.

3. Access Control

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user.

Password Selection Strategies

Four basic techniques are in use:

- User education
- Computer-generated passwords
- Reactive password checking
- Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines

Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down

A **reactive password** checking strategy is one in which the system periodically runs its own password cracker to find guessable passwords.

The most promising approach to improved password security is a **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be

guessed in a dictionary attack.

The first approach is a simple system for rule enforcement. For example, the following rules could be enforced:

- All passwords must be at least eight characters long.
- In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks. These rules could be coupled with advice to the user. Although this approach is superior to simply educating users, it may not be sufficient to thwart password crackers. This scheme alerts crackers as to which passwords not to try but may still make it possible to do password cracking.

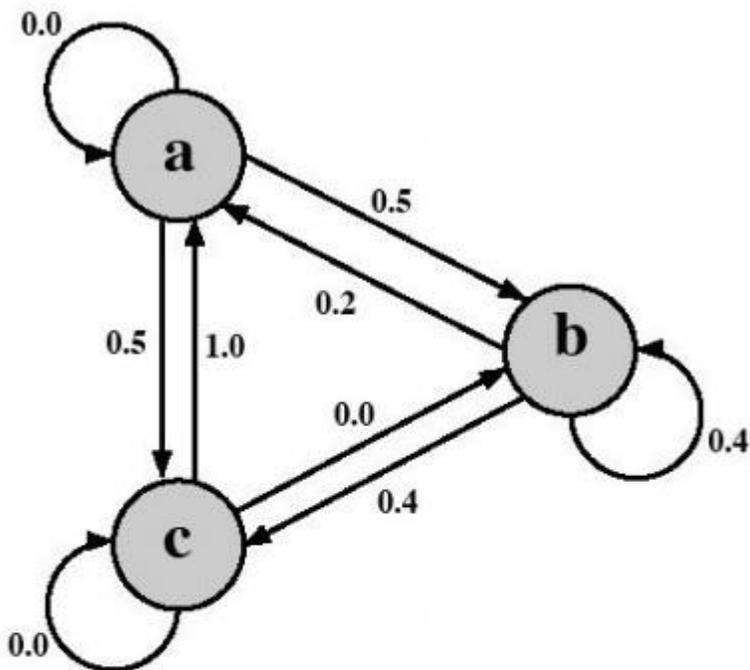
Another possible procedure is simply to compile a large dictionary of possible "bad" passwords. When a user selects a password, the system checks to make sure that it is not on the disapproved list.

There are two problems with this approach:

- Space: The dictionary must be very large to be effective..
- Time: The time required to search a large dictionary may itself be large

Two techniques for developing an effective and efficient proactive password checker that is based on rejecting words on a list show promise. One of these develops a Markov model for the generation of guessable passwords. This model shows a language consisting of an alphabet of three characters. The state of the system at any time is the identity of the most recent letter. The value on the transition from one state to another represents the probability that one letter follows another. Thus, the probability that the next letter is b, given that the current letter is a, is 0.5.

In general, a Markov model is a quadruple $[m, A, T, k]$, where m is the number of states in the model, A is the state space, T is the matrix of transition probabilities, and k is the order of the model. For a k th-order model, the probability of making a transition to a particular letter depends on the previous k letters that have been generated.



$M = \{3, \{a, b, c\}, T, 1\}$ where

$$T = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{bmatrix}$$

e.g., string probably from this language: **abbcaacaba**

e.g., string probably not from this language: **aaccbbaaa**

The authors report on the development and use of a second-order model. To begin, a dictionary of guessable passwords is constructed. Then the transition matrix is calculated as follows:

1. Determine the frequency matrix f , where $f(i, j, k)$ is the number of occurrences of the trigram consisting of the i th, j th, and k th character. For example, the password **parsnips** yields the trigrams **par**, **ars**, **rsn**, **sni**, **nip**, and **ips**.
2. For each bigram ij , calculate $f(i, j, \infty)$ as the total number of trigrams beginning with ij . For example, $f(a, b, \infty)$ would be the total number of trigrams of the form **aba**, **abb**, **abc**, and so on.
3. Compute the entries of T as follows:

$$T(i,j,k) = f(i, j, k) / f(i, j, \infty)$$

The result is a model that reflects the structure of the words in the dictionary.

A quite different approach has been reported by Spafford. It is based on the use of a Bloom filter. To begin, we explain the operation of the Bloom filter. A Bloom filter of order k consists of a set of k independent hash functions $H_1(x), H_2(x), \dots, H_k(x)$, where each function maps a password into a hash value in the range 0 to $N - 1$. That is,

$$H_i(X_j) = y \quad 1 \leq i \leq k; \quad 1 \leq j \leq D; \quad 0 \leq y \leq N - 1$$

where

X_j = j th word in password dictionary

D = number of words in password dictionary

The following procedure is then applied to the dictionary:

· A hash table of N bits is defined, with all bits initially set to 0.

· For each password, its k hash values are calculated, and the corresponding bits in the hash table are set to 1. Thus, if $H_i(X_j) = 67$ for some (i, j) , then the sixty-seventh bit of the hash table is set to 1; if the bit already has the value 1, it

remains at 1.

When a new password is presented to the checker, its k hash values are calculated. If all the corresponding bits of the hash table are equal to 1, then the password is rejected.

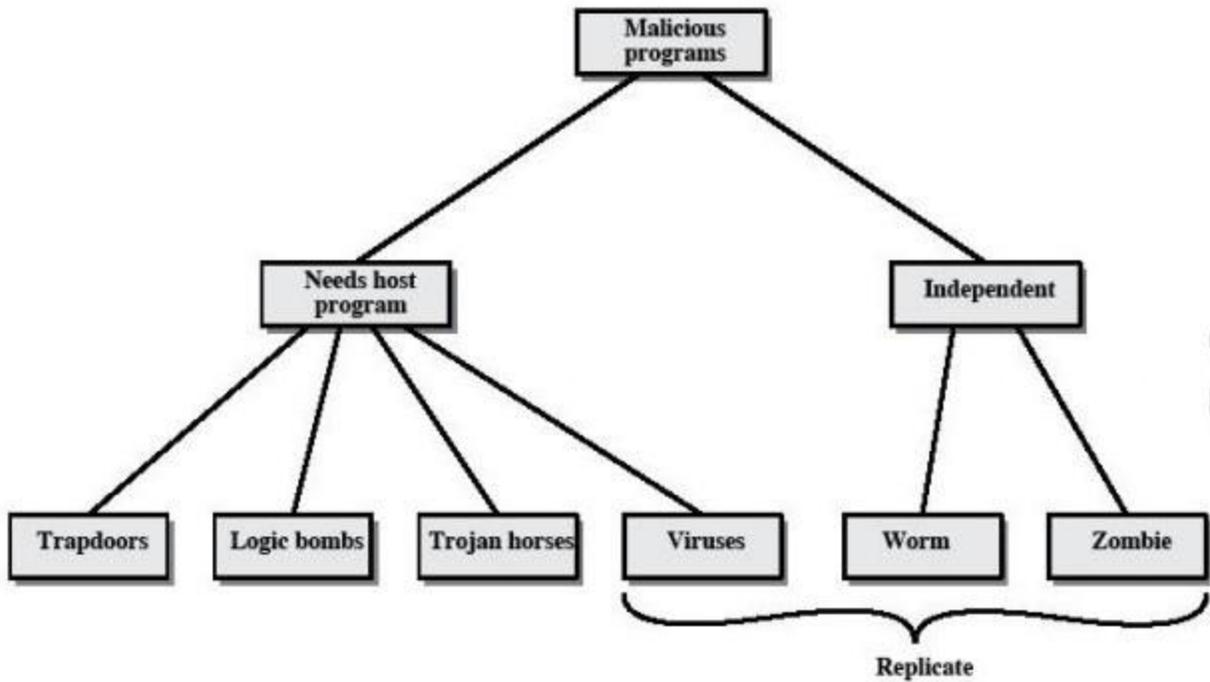
VIRUSES AND RELATED THREATS

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

1. Malicious Programs

Malicious software can be divided into two categories:

- Those that need a host program
- Those that are independent.



he former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

Name	Description
Virus	Attaches itself to a program and propagates copies of itself to other programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor (trapdoor)	Program modification that allows unauthorized access to functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-router	Malicious hacker tools used to break into new machines remotely
Kit (virus generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

2. The Nature of Viruses

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

- **Dormant phase:** The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
- **Propagation phase:** The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
- **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.
- **Execution phase:** The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

3. Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

An infected program begins with the virus code and works as follows.

The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.

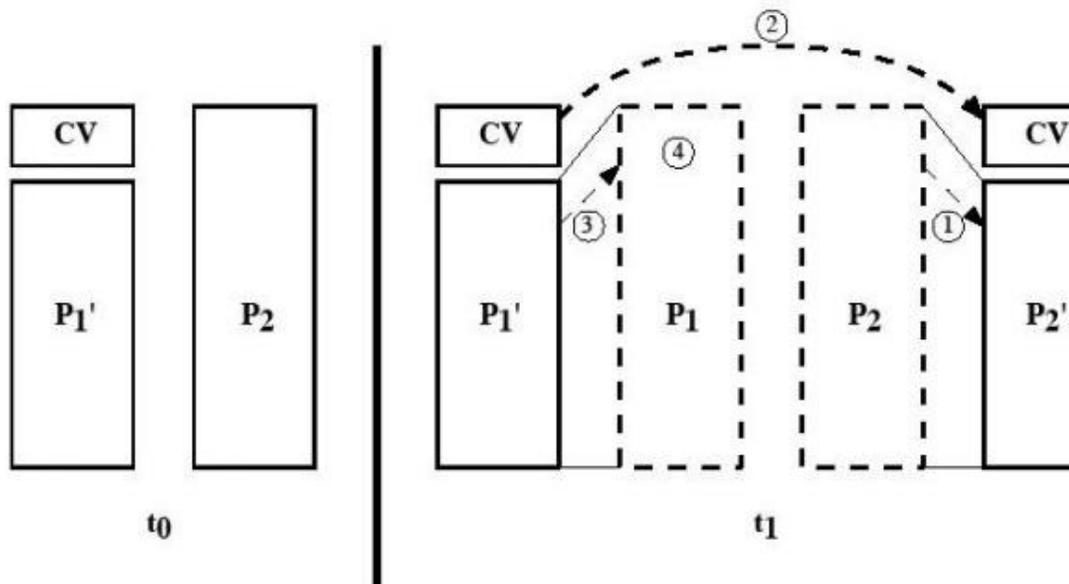
When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.

This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.

Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length. The key lines in this virus are numbered. We assume that program P_1 is infected with the virus CV . When this program is invoked, control passes to its virus, which performs the following steps:

1. For each uninfected file P_2 that is found, the virus first compresses that file to produce P_2' , which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P_1' , is uncompressed.
4. The uncompressed original program is executed.



In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

4. Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of iron and write all one's own system and application programs, one is vulnerable.

VIRUS COUNTERMEASURES

Antivirus Approaches

The ideal solution to the threat of viruses is prevention: The next best approach is to be able to do the following:

- **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.
- **Identification:** Once detection has been achieved, identify the specific virus that has infected a program.
- **Removal:** Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.

There are four generations of antivirus software:

- First generation: simple scanners
- Second generation: heuristic scanners
- Third generation: activity traps
- Fourth generation: full-featured protection

A **first-generation scanner** requires a virus signature to identify a virus.. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

A **second-generation scanner** does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the virus cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the virus is prevented from adjusting the program to produce the same hash code as before.

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

Fourth-generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures.

Advanced Antivirus Techniques

More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight two of the most important.

Generic Decryption

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds . In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

- **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

- **Virus signature scanner:** A module that scans the target code looking for known virus signatures.

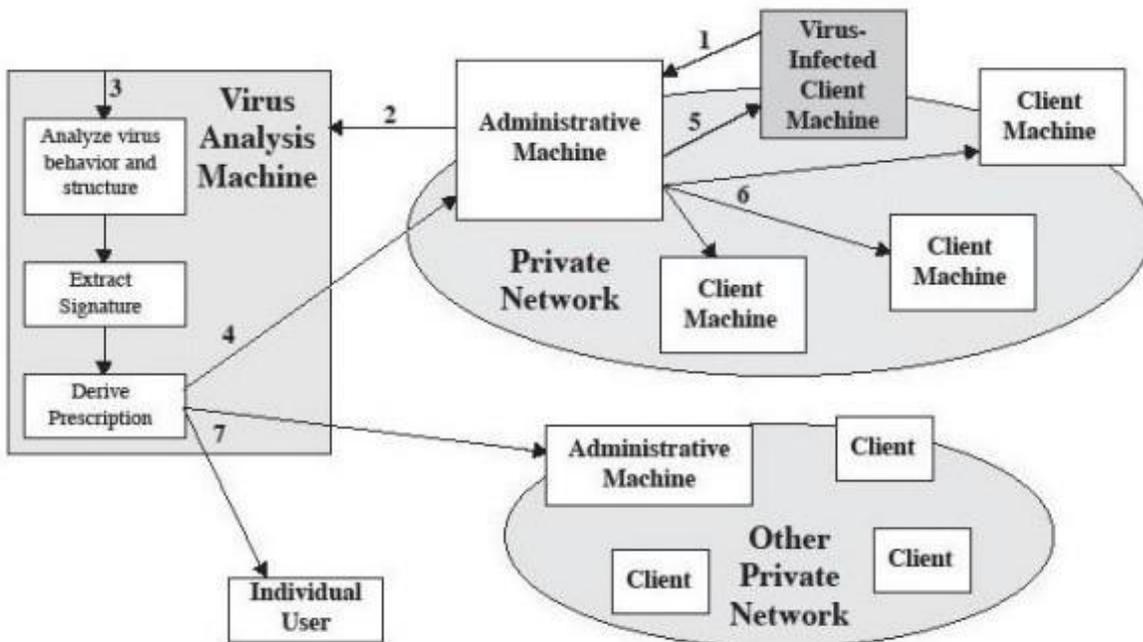
- **Emulation control module:** Controls the execution of the target code.

Digital Immune System

The digital immune system is a comprehensive approach to virus protection developed by IBM]. The motivation for this development has been the rising threat of Internet-based virus propagation. Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:

Integrated mail systems: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

Mobile-program systems: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.



- A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.

- The administrative machine encrypts the sample and sends it to a central virus analysis machine.

- This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored. The virus analysis machine then produces a prescription for identifying and removing the virus.
- The resulting prescription is sent back to the administrative machine.
- The administrative machine forwards the prescription to the infected client.
- The prescription is also forwarded to other clients in the organization.
- Subscribers around the world receive regular antivirus updates that protect them from the new virus.

The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains. By constantly analyzing and monitoring the viruses found in the wild, it should be possible to continually update the digital immune software to keep up with the threat.

Behavior-Blocking Software

Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. Monitored behaviors can include the following:

- Attempts to open, view, delete, and/or modify files;
- Attempts to format disk drives and other unrecoverable disk operations;
- Modifications to the logic of executable files or macros;
- Modification of critical system settings, such as start-up settings;
- Scripting of e-mail and instant messaging clients to send executable content; and
- Initiation of network communications.

If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established antivirus detection techniques as fingerprinting or heuristics.

Distributed denial of service (DDoS) attack

A distributed denial-of-service (DDoS) attack is an attack in which multiple compromised computer systems attack a target, such as a server, website or other network resource, and cause a denial of service for users of the targeted resource. The flood of incoming messages, connection requests or malformed packets to the target system forces it to slow down or even crash and shut down, thereby denying service to legitimate users or systems.

DDoS attacks have been carried out by diverse threat actors, ranging from individual criminal hackers to organized crime rings and government agencies. In certain situations, often ones related to poor coding, missing patches or generally unstable systems, even legitimate requests to target systems can result in DDoS-like results.

How DDoS attacks work

In a typical DDoS attack, the assailant begins by exploiting a vulnerability in one computer system and making it the DDoS master. The attack master system identifies other vulnerable systems and gains control over them by either infecting the systems with malware or through bypassing the authentication controls (i.e., guessing the default password on a widely used system or device).

A computer or networked device under the control of an intruder is known as a zombie, or bot. The attacker creates what is called a command-and-control server to command the network of bots, also called a botnet. The person in control of a botnet is sometimes referred to as the botmaster (that term has also historically been used to refer to the first system "recruited" into a botnet because it is used to control the spread and activity of other systems in the botnet).

Botnets can be comprised of almost any number of bots; botnets with tens or hundreds of thousands of nodes have become increasingly common, and there may not be an upper limit to their size. Once the botnet is assembled, the attacker can use the traffic generated by the compromised devices to flood the target domain and knock it offline.

Types of DDoS attacks

There are three types of DDoS attacks. Network-centric or volumetric attacks overload a targeted resource by consuming available bandwidth with packet floods. Protocol attacks target network layer or transport layer protocols using flaws in the protocols to overwhelm targeted resources. And application layer attacks overload application services or databases with a high volume of application calls. The inundation of packets at the target causes a denial of service.

While it is clear that the target of a DDoS attack is a victim, there can be many other victims in a typical DDoS attack, including the owners of the systems used to execute the attack. Although the owners of infected computers are typically unaware their systems have been compromised, they are nevertheless likely to suffer a degradation of service during a DDoS attack.

Internet of things and DDoS attacks

While the things comprising the internet of things (IoT) may be useful to legitimate users, in some cases, they are even more helpful to DDoS attackers. The devices connected to IoT include any appliance into which some computing and networking capacity has been built, and, all too often, these devices are not designed with security in mind.

Devices connected to the IoT expose large attack surfaces and display minimal attention to security best practices. For example, devices are often shipped with hard-coded authentication credentials for system administration, making it simple for attackers to log in to the devices. In some cases, the authentication credentials cannot be changed. Devices also often ship without the capability to upgrade or patch device software, further exposing them to attacks that leverage well-known vulnerabilities.

Internet of things botnets are increasingly being used to wage massive DDoS attacks. In 2016, the Mirai botnet was used to attack the domain name service provider Dyn, based in Manchester, N.H.; attack volumes were measured at over 600 Gbps. Another late 2016 attack unleashed on OVH, the French hosting firm, peaked at more than 1 Tbps.

DDoS defense and prevention

DDoS attacks can create significant business risks with lasting effects. Therefore, it is important for IT and security administrators and managers, as well as their business executives, to understand the threats, vulnerabilities and risks associated with DDoS attacks.

Being on the receiving end of a DDoS attack is practically impossible to prevent. However, the business impact of these attacks can be minimized through some core information security practices, including performing ongoing security assessments to look for -- and resolve -- denial of service-related vulnerabilities and using network security controls, including services from cloud-based vendors specializing in responding to DDoS attacks.

In addition, solid patch management practices, email phishing testing and user awareness, and proactive network monitoring and alerting can help minimize an organization's contribution to DDoS attacks across the internet.

Firewall design principles

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. The alternative, increasingly accepted, is the firewall.

The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Firewall characteristics:

- All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall.

- Various configurations are possible.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- Various types of firewalls are used, which implement various types of security policies.
- The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site's security policy is as follows:

- Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.
- Direction control – determines the direction in which particular service request may be initiated and allowed to flow through the firewall.
- User control – controls access to a service according to which user is attempting to access it.
- Behavior control – controls how particular services are used.

Capabilities of firewall

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

A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

A firewall is a convenient platform for several internet functions that are not security related.

A firewall can serve as the platform for IPsec.

Limitations of firewall

- The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.
- The firewall does not protect against internal threats. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
- The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

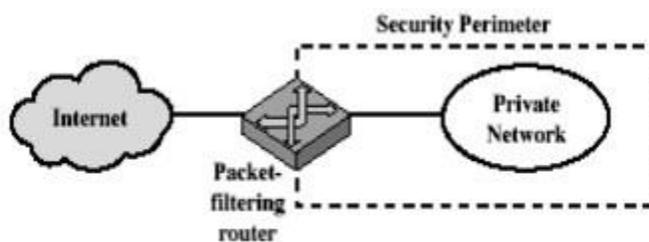
Types of firewalls

There are 3 common types of firewalls.

1. Packet filters
2. Application-level gateways
3. Circuit-level gateways

Packet filtering router

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

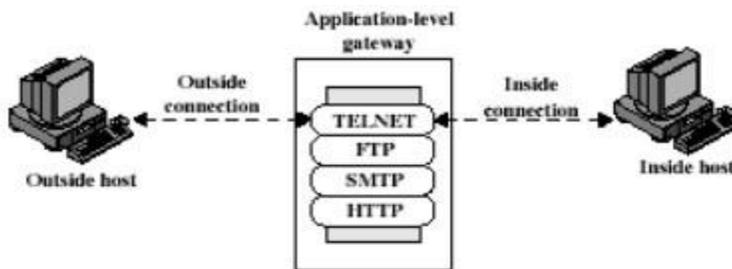


(a) Packet-filtering router

Application level gateway

An Application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

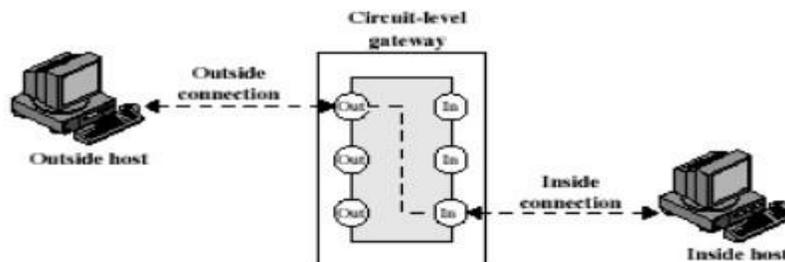
Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.



(b) Application-level gateway

Circuit level gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed. A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.



(c) Circuit-level gateway

Trusted System :

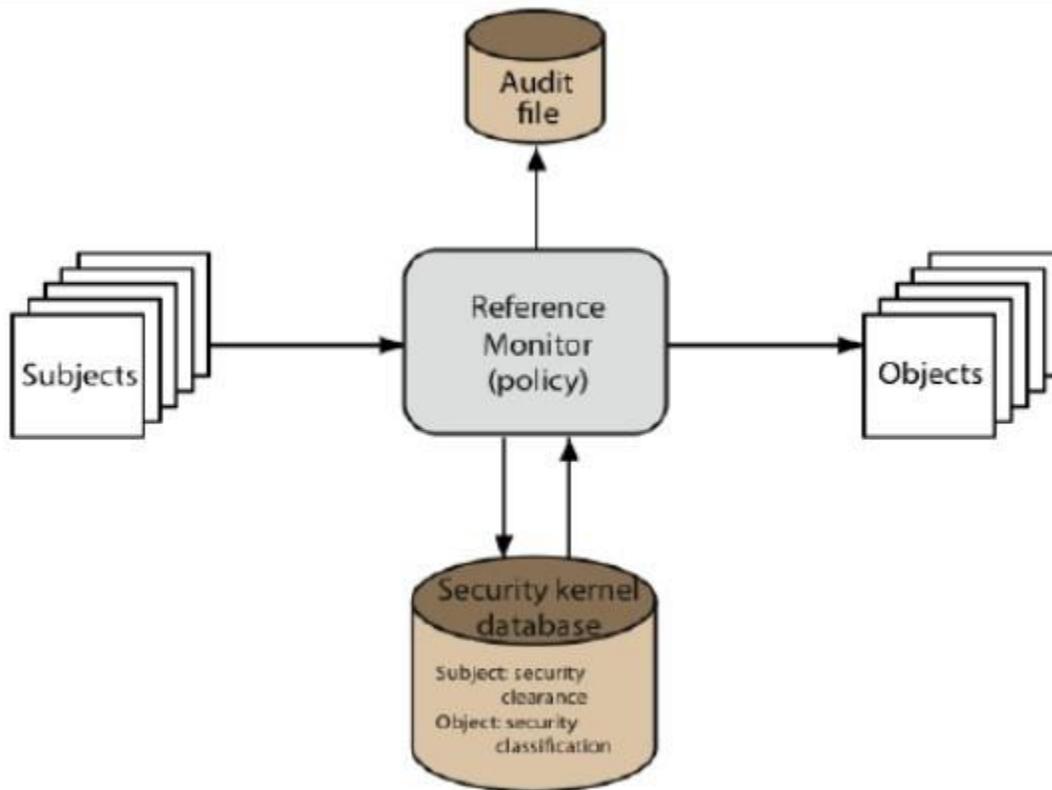
A widely applicable approach for protection of data and resources is based on levels of security. This is commonly found in military, where information is categorized as unclassified (U), confidential (C), secret (S), top secret (TS), or beyond. This concept is equally applicable in other areas, where information can be organized into categories and users can be granted clearances to access certain categories of data. When multiple categories or levels of data are defined, the requirement is referred to as **multilevel security**. The general statement of the requirement for multilevel security is that a subject at a high level may not convey information to a subject at a lower or non-comparable level unless that flow accurately reflects the will of an authorized user. For implementation purposes, this requirement is in two parts and is simply stated. A multilevel secure system must enforce the following:

- **No read-up:** A subject can only read an object of less or equal security level. This is referred to in the literature as the **simple security property**
- **No write-down:** A subject can write into an object of greater or equal security level. This is referred to as the ***-property** (pronounced star property)

These two rules, if properly enforced, provide multilevel security. The Reference Monitor concept was introduced as an ideal to achieve controlled sharing. The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of security parameters of the subject and object. The reference monitor has access to a file, known as the security kernel database that lists the access privilege (security clearance) of each subject and the protection attributes (classification level) of each object. The reference monitor enforces the security rules (no read-up, no write-down). A combination of hardware, software, and firmware that implements the Reference Monitor concept is called the *Reference Validation Mechanism* and has the following properties:

- ❖ *Complete mediation:* The Reference Validation Mechanism must always be invoked.
- ❖ *Isolation:* The Reference Validation Mechanism must be tamperproof.
- ❖ *Verifiability:* The Reference Validation Mechanism must be small enough to be subjected to analysis and tests to ensure that it is correct.

The above mentioned requirements are very stiff. Complete mediation requires that every access to data within main memory and on disk and tape must be mediated. Though pure software implementation is not practical, solution is at least partly hardware implementation. The requirement for isolation means that it must not be possible for an attacker, no matter how clever, to change the logic of the reference monitor or the contents of the security kernel database. Finally, the requirement for mathematical proof is formidable for something as complex as a general-purpose computer. A system that can provide such verification is referred to as a **trusted system**.



A final element in the Reference Monitor concept is an audit file. Important security events, such as detected security violations and authorized changes to the security kernel database, are stored in the audit file.