19PCSC203 - Cryptography and Network Security

Unit-1: Introduction - Security trends – Legal, Ethical and Professional Aspects of Security, Need for Security at Multiple levels, Security Policies – Model of network security – Security attacks, services and mechanisms – OSI security architecture – Classical encryption techniques: substitution techniques, transposition techniques, steganography- Foundations of modern cryptography: perfect security – information theory – product cryptosystem – cryptanalysis.

Unit-2: Symmetric Encryption and Message Confidentiality - Symmetric Encryption Principles, Symmetric Block Encryption Algorithms, Stream Ciphers and RC4, Chipher Block Modes of Operation, Location of Encryption Devices, Key Distribution. Public-key Cryptography and Message Authentication: Approaches to Message Authentication, Secure Hash Functions and HMAC, Public-Key Cryptography Principles, Public-Key Cryptography Algorithms, Digital Signatures, Key Management.

Unit-3: Authentication Applications - Kerberos, x.509 Authentication Service, Public-Key Infrastructure. Electronic Mail Security: Pretty Good Privacy (PGP), S/MIME.

Unit-4: IP Security - IP Security Over view, IP Security Architecture, Authentication Header, Encapsulating Security Payload, Combining Security Associations. Web Security: Web Security Considerations, Secure Socket Layer(SSL) and Transport Layer Security(TLS), Secure Electronic Transaction(SET).Network Management Security: Basic Concepts of SNMP, SNMPv1 Community Facility, SNMPv3.

Unit-5: Intruders - Intruders, Intrusion Detection, Password Management. Malicious Software: Virus and Related Threats, Virus Countermeasures, Distributed Denial of Service Attacks. Firewalls: Firewall Design Principles, Trusted Systems, Common Criteria for Information Technology Security Evaluation.

Text Books:

1. Behrouz A. Ferouzan, “Cryptography & Network Security”, Tata Mc Graw Hill, 2007, Reprint 2015.
2. Stallings William, “Cryptography and Network Security - Principles and Practice 2017.
3. William Stallings, “Network Security Essentials Applications and Standards”, Third Edition, Pearson Education, 2008.

Reference Books

1. Man Young Rhee, “Internet Security: Cryptographic Principles”, “Algorithms And Protocols”, Wiley Publications, 2003.
2. Charles Pfleeger, “Security In Computing”, 4th Edition, Prentice Hall Of India, 2006.
3. Ulysess Black, “Internet Security Protocols”, Pearson Education Asia, 2000.
4. Charlie Kaufman And Radia Perlman, Mike Speciner, “Network Security, Second Edition, Private Communication In Public World”, PHI 2002.

UNIT - I

INTRODUCTION

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

Basic Concepts

Cryptography The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

Plaintext The original intelligible message

Cipher text The transformed message

Cipher An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

Key Some critical information used by the cipher, known only to the sender & receiver

Encipher (encode) The process of converting plaintext to cipher text using a cipher and a key

Decipher (decode) the process of converting cipher text back into plaintext using a cipher and a key

Cryptanalysis The study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called **code breaking**

Cryptology Both cryptography and cryptanalysis

Code An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Cryptography

Cryptographic systems are generally classified along 3 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.

Legal, Ethical and Professional Aspects of Security

Law and Ethics in Information Security

Laws are rules that mandate or prohibit certain behavior in society; they are drawn from **ethics**, which define socially acceptable behaviors. The key difference between laws and ethics is that laws carry the sanctions of a governing authority and ethics do not. Ethics in turn are based on **Cultural mores.**

Ethical Concepts in Information Security

Cultural Differences in Ethical Concepts

· Differences in cultures cause problems in determining what is ethical and what is not ethical

· Studies of ethical sensitivity to computer use reveal different nationalities have different perspectives

· Difficulties arise when one nationality's ethical behavior contradicts that of another national group

Ethics and Education

Employees must be trained and kept aware of a number of topics related to information security, not the least of which is the expected behaviors of an ethical employee

This is especially important in areas of information security, as many employees may not have the formal technical training to understand that their behavior is unethical or even illegal

Proper ethical and legal training is vital to creating an informed, well prepared, and low-risk system user

Deterrence to Unethical and Illegal Behavior

ü Deterrence - preventing an illegal or unethical activity
ü Laws, policies, and technical controls are all examples of deterrents
ü Laws and policies only deter if three conditions are present:

Fear of penalty
Probability of being caught
Probability of penalty being administered

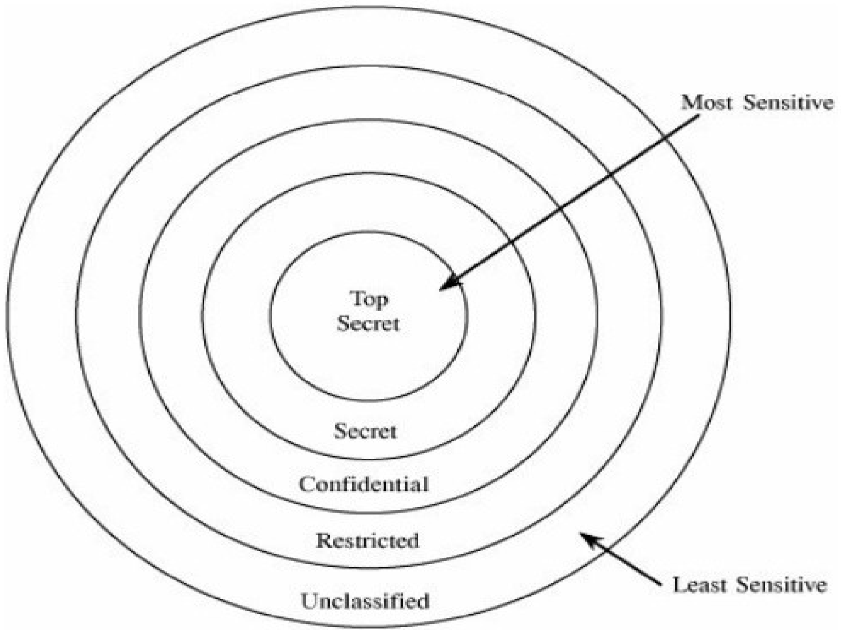
SECURITY POLICIES

To know that an operating system maintains the security we expect, we must be able to state its security policy. A **security policy** is a statement of the security we expect the system to enforce. An operating system (or any other piece of a trusted system) can be trusted only in relation to its security policy; that is, to the security needs the system is expected to satisfy.

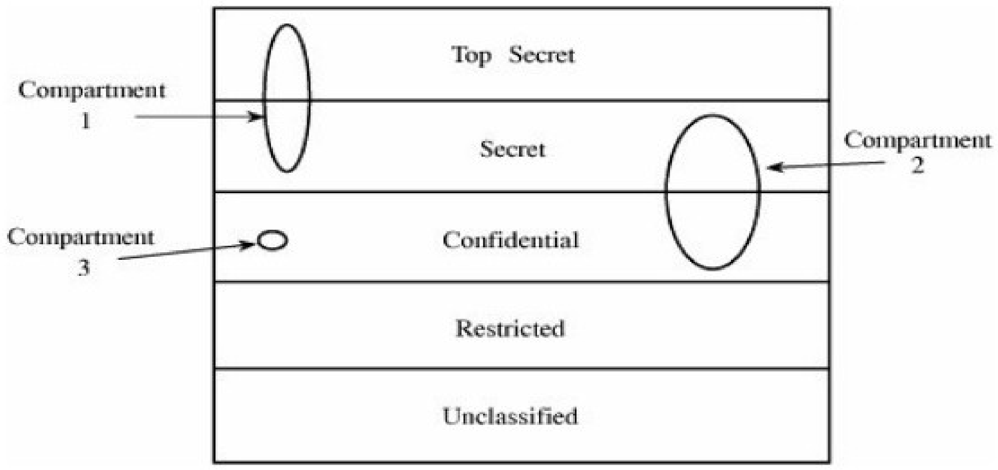
We begin our study of security policy by examining military security policy because it has been the basis of much trusted operating system development and is fairly easy to state precisely. Then, we move to security policies that commercial establishments might adopt.

Military Security Policy

Military security policy is based on protecting classified information. Each piece of information is ranked at a particular sensitivity level, such as unclassified, restricted, confidential, secret, or top secret. The ranks or levels form a hierarchy, and they reflect an increasing order of sensitivity, as shown in Figure. That is, the information at a given level is more sensitive than the information in the level below it and less sensitive than in the level above it. For example, restricted information is more sensitive than unclassified but less sensitive than confidential. We can denote the sensitivity of an object O by $rank_O$. In the rest of this chapter we assume these five sensitivity levels.

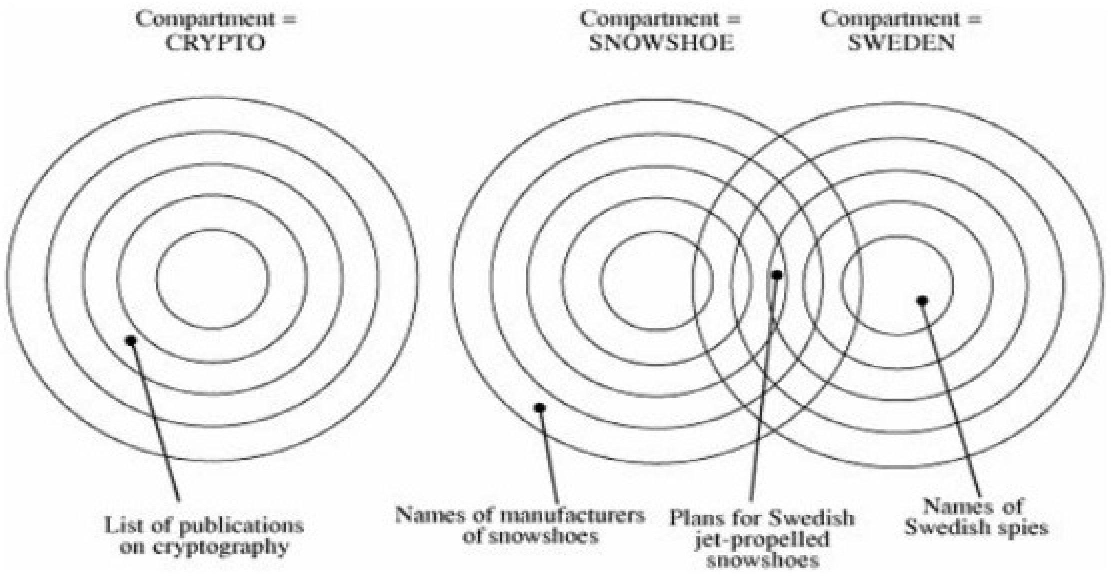


Information access is limited by the **need-to-know** rule: Access to sensitive data is allowed only to subjects who need to know those data to perform their jobs. Each piece of classified information may be associated with one or more projects, called **compartments**, describing the subject matter of the information. For example, the alpha project may use secret information, as may the beta project, but staff on alpha do not need access to the information on beta. In other words, both projects use secret information, but each is restricted to only the secret information needed for its particular project. In this way, compartments help enforce need-to-know restrictions so that people obtain access only to information that is relevant to their jobs. A compartment may include information at only one sensitivity level, or it may cover information at several sensitivity levels. The relationship between compartments and sensitivity levels is shown in Figure.



We can assign names to identify the compartments, such as snowshoe, crypto, and Sweden. A single piece of information can be coded with zero, one, two, or more compartment names, depending on the categories to which it relates. The association of information and compartments is shown in Figure. For example, one piece of information may be a list of publications on cryptography, whereas another may describe development of snowshoes in Sweden. The compartment of this first piece of information is {crypto}; the second is {snowshoe, Sweden}.

Figure 5-3. Association of Information and Compartments.


Compartment = CRYPTO
Compartment = SNOWSHOE
Compartment = SWEDEN
List of publications on cryptography
Names of manufacturers of snowshoes
Plans for Swedish jet-propelled snowshoes
Names of Swedish spies

The combination <rank; compartments> is called the class or **classification** of a piece of information. By designating information in this way, we can enforce need-to-know both by security level and by topic.

A person seeking access to sensitive information must be cleared. A **clearance** is an indication that a person is trusted to access information up to a certain level of sensitivity and that the person needs to know certain categories of sensitive information. The clearance of a subject is expressed as a combination <rank; compartments>. This combination has the same form as the classification of a piece of information.

$s \leq o$ if and only if

$rank_s \leq rank_o$ and

$compartments_s \subseteq compartments_o$

the clearance level of the subject is at least as high as that of the information and
the subject has a need to know about all compartments for which the information is classified These conditions are equivalent to saying that the subject dominates the object.

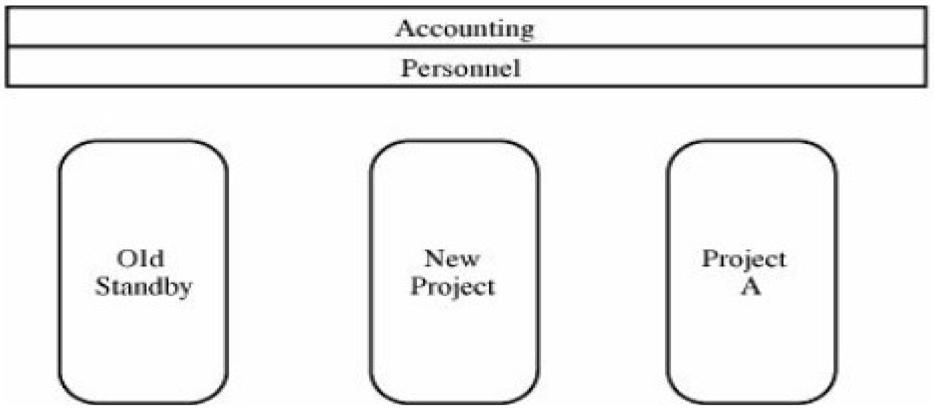
To see how the dominance relation works, consider the concentric circles in Figure. According to the relationships depicted there, information classified as <secret;{Sweden}> could be read by someone cleared for access to <top secret;{Sweden}> or <secret;{Sweden, crypto}>, but not by someone with a <top secret;{crypto}> clearance or someone cleared for <confidential;{Sweden}> or <secret;{France} >.

Military security enforces both sensitivity requirements and need-to-know requirements. Sensitivity requirements are known as **hierarchical** requirements because they reflect the hierarchy of sensitivity levels; need-to-know restrictions are **nonhierarchical** because compartments do not necessarily reflect a hierarchical structure. This combinational model is appropriate for a setting in which access is rigidly controlled by a central authority. Someone, often called a security officer, controls clearances and classifications, which are not generally up to individuals to alter.

Commercial Security Policies

Commercial enterprises have significant security concerns. They worry that industrial espionage will reveal information to competitors about new products under development. Likewise, corporations are often eager to protect information about the details of corporate finance. So even though the commercial world is usually less rigidly and less hierarchically structured than the military world, we still find many of the same concepts in commercial security policies. For example, a large organization, such as a corporation or a university, may be divided into groups or departments, each responsible for a number of disjoint projects. There may also be some corporate-level responsibilities, such as accounting and personnel activities. Data items at any level may have different degrees of sensitivity, such as public, proprietary, or internal; here, the names may vary among organizations, and no universal hierarchy applies.

Let us assume that public information is less sensitive than proprietary, which in turn is less sensitive than internal. Projects and departments tend to be fairly well separated, with some overlap as people work on two or more projects. Corporate-level responsibilities tend to overlie projects and departments, as people throughout the corporation may need accounting or personnel data. However, even corporate data may have degrees of sensitivity. Projects themselves may introduce a degree of sensitivity: Staff members on project old-standby have no need to know about project new-product, while staff members on new-product may have access to all data on old-standby. For these reasons, a commercial layout of data might look like Figure.



Two significant differences exist between commercial and military information security. First, outside the military, there is usually no formalized notion of clearances: A person working on a commercial project does not require approval for project MARS access by a central security officer. Typically, an employee is not conferred a different degree of trust by being allowed access to internal data. Second, because there is no formal concept of a clearance, the rules for allowing access are less regularized. For example, if a senior manager decides that a person needs access to a piece of MARS internal data, the manager will instruct someone to allow the access, either one-time or continuing. Thus, there is no dominance function for most commercial information access because there is no formal concept of a commercial clearance.

So far, much of our discussion has focused only on read access, which addresses confidentiality in security. In fact, this narrow view holds true for much of the existing work in computer security. However, integrity and availability are at least as important as confidentiality in many instances. Policies for integrity and availability are significantly less well formulated than those for confidentiality, in both military and commercial realms. In the two examples that follow, we explore some instances of integrity concerns.

ClarkWilson Commercial Security Policy

In many commercial applications, integrity can be at least as important as confidentiality. The correctness of accounting records, the accuracy of legal work, and the proper timing of medical treatments are the essence of their fields. Clark and Wilson proposed a policy for what they call *well-formed transactions*, which they assert are as important in their field as is confidentiality in a military realm.

To see why, consider a company that orders and pays for materials. A representation of the procurement process might be this: A purchasing clerk creates an order for a supply, sending copies of the order to both the supplier and the receiving department.

The supplier ships the goods, which arrive at the receiving department. A receiving clerk checks the delivery, ensures that the correct quantity of the right item has been received, and signs a delivery form. The delivery form and the original order go to the accounting department.

The supplier sends an invoice to the accounting department. An accounting clerk compares the invoice with the original order (as to price and other terms) and the delivery form (as to quantity and item) and issues a check to the supplier.

The sequence of activities is important. A receiving clerk will not sign a delivery form without already having received a matching order (because suppliers should not be allowed to ship any quantities of any items they want and be paid), and an accounting clerk will not issue a check without already having received a matching order and delivery form (because suppliers should not be paid for goods not ordered or received). Furthermore, in most cases, both the order and the delivery form must be signed by authorized individuals. Performing the steps in order, performing exactly the steps listed, and authenticating the individuals who perform the steps constitute a well-formed transaction. The goal of the ClarkWilson policy is to maintain consistency between the internal data and the external (users') expectations of those data.

Clark and Wilson present their policy in terms of **constrained data items**, which are processed by **transformation procedures**. A transformation procedure is like a monitor in that it performs only particular operations on specific kinds of data items; these data items are manipulated only by transformation procedures. The transformation procedures maintain the integrity of the data items by validating the processing to be performed. Clark and Wilson propose defining the policy in terms of **access triples**: $<userID, TP_i, \{CDI_j, CDI_k, ...\}>$,

combining a transformation procedure, one or more constrained data items, and the identification of a user who is authorized to operate on those data items by means of the transaction procedure.

Separation of Duty

A second commercial security policy involves separation of responsibility. Clark and Wilson [CLA87] raised this issue in their analysis of commercial security requirements, and Lee [LEE88] and Nash and Poland [NAS90] added to the concept.

To see how it works, we continue our example of a small company ordering goods. In the company, several people might be authorized to issue orders, receive goods, and write checks. However, we would not want the same person to issue the order, receive the goods, and write the check, because there is potential for abuse. Therefore, we might want to establish a policy that specifies that three separate individuals issue the order, receive the goods, and write the check, even though any of the three might be authorized to do any of these tasks. This required division of responsibilities is called **separation of duty**.

Separation of duty is commonly accomplished manually by means of dual signatures. Clark and Wilson triples are "stateless," meaning that a triple does not have a context of prior operations; triples are incapable of passing control information to other triples. Thus, if one person is authorized to perform operations TP_1 and TP_2, the Clark and Wilson triples cannot prevent the same person from performing both TP_1 and TP_2 on a given data item. However, it is quite easy to implement distinctness if it is stated as a policy requirement.

Chinese Wall Security Policy

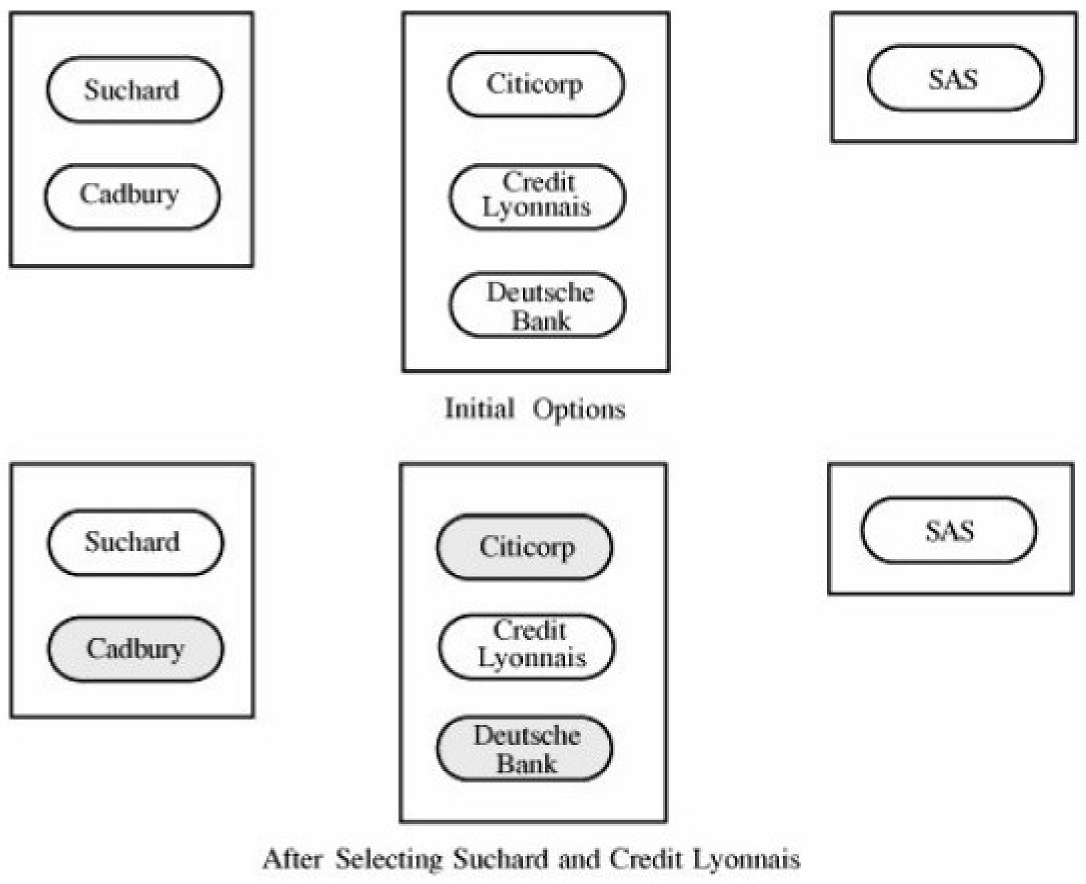
Brewer and Nash [BRE89] defined a security policy called the Chinese Wall that reflects certain commercial needs for information access protection. The security requirements reflect issues relevant to those people in legal, medical, investment, or accounting firms who might be subject to conflict of interest. A conflict of interest exists when a person in one company can obtain sensitive information about people, products, or services in competing companies.

The security policy builds on three levels of abstraction.

Objects. At the lowest level are elementary objects, such as files. Each file contains information concerning only one company.
Company groups. At the next level, all objects concerning a particular company are grouped together.
Conflict classes. At the highest level, all groups of objects for competing companies are clustered.

With this model, each object belongs to a unique company group, and each company group is contained in a unique conflict class. A conflict class may contain one or more company groups. For example, suppose you are an advertising company with clients in several fields: chocolate companies, banks, and airlines. You might want to store data on chocolate companies Suchard and Cadbury; on banks Citicorp, Deutsche Bank, and Credit Lyonnais; and on airline SAS. You want to prevent your employees from inadvertently revealing information to a client about that client's competitors, so you establish the rule that no employee will know sensitive information about competing companies. Using the Chinese Wall hierarchy, you would form six company groups (one for each company) and three conflict classes: {Suchard, Cadbury}, {Citicorp, Deutsche Bank, Credit Lyonnais}, and {SAS}.

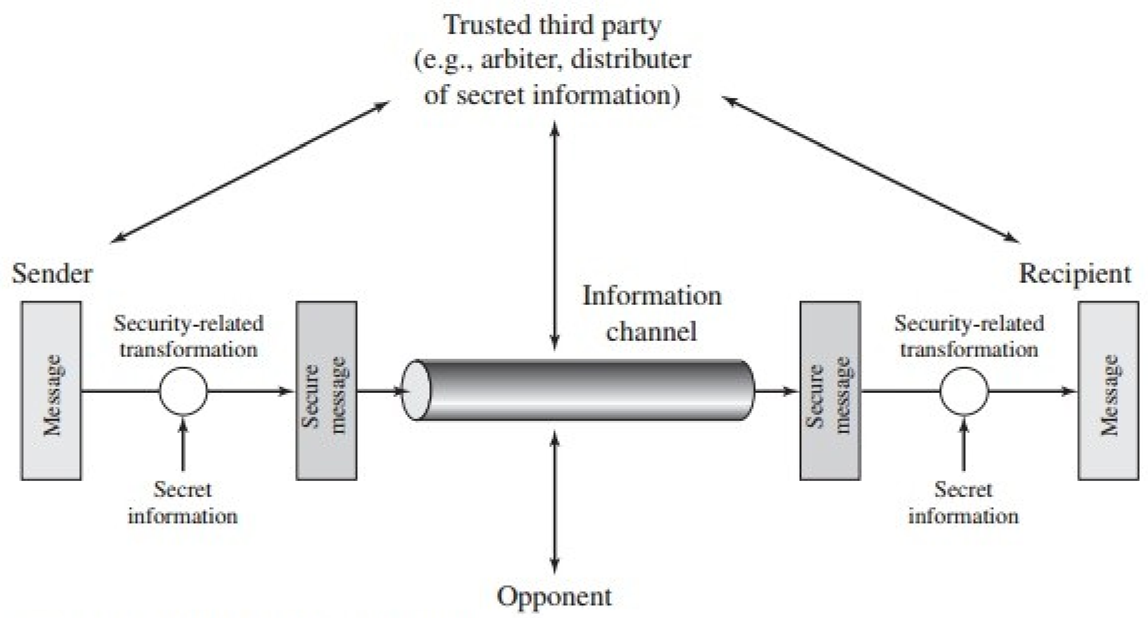
The hierarchy guides a simple access control policy: A person can access any information as long as that person has never accessed information from a different company in the same conflict class. That is, access is allowed if either the object requested is in the same company group as an object that has previously been accessed or the object requested belongs to a conflict class that has never before been accessed. In our example, initially you can access any objects. Suppose you read from a file on Suchard. A subsequent request for access to any bank or to SAS would be granted, but a request to access Cadbury files would be denied. Your next access, of SAS data, does not affect future accesses. But if you then access a file on Credit Lyonnais, you will be blocked from future accesses to Deutsche Bank or Citicorp. From that point on, as shown in Figure 5-5, you can access objects only concerning Suchard, SAS, Credit Lyonnais, or a newly defined conflict class.



The Chinese Wall is a commercially inspired confidentiality policy. It is unlike most other commercial policies, which focus on integrity. It is also interesting because access permissions change dynamically: As a subject accesses some objects, other objects that would previously have been accessible are subsequently denied.

A MODEL FOR NETWORK SECURITY

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



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

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.
This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of the secret information.

4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Parts One through Five of this book concentrate on the types of security mecha-nisms and services that fit into the model shown in Figure. However, there are other security-related situations of interest that do not neatly fit this model but are consid-ered in this book. A general model of these other situations is illustrated by Figure 1.5, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain

(e.g., obtaining credit card numbers or performing illegal money transfers).

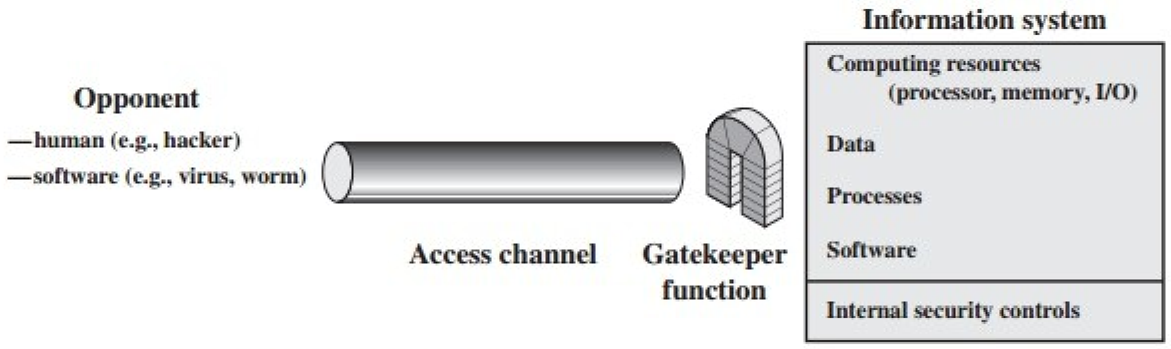


Figure 1.5 Network Access Security Model

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

- **Information access threats:** Intercept or modify data on behalf of users who should not have access to that data.

- **Service threats:** Exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic con-cealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

THE OSI SECURITY ARCHITECTURE

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for computer and network security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. This is difficult enough in a centralized data processing environment; with the use of local and wide area networks, the problems are compounded.

ITU-T4 Recommendation X.800, Security Architecture for OSI, defines such a systematic approach. The OSI security architecture is useful to managers as a way of organizing the task of providing security. Furthermore, because this architecture was developed as an international standard, computer and communications vendors have developed security features for their products and services that relate to this structured definition of services and mechanisms.

For our purposes, the OSI security architecture provides a useful, if abstract, overview of many of the concepts that this book deals with.The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as

Security attack - Any action that compromises the security of information owned by an organization.
Security mechanism - A mechanism that is designed to detect, prevent or recover from a security attack.
Security service - A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service.

SECURITY ATTACKS

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. This information might be useful in guessing the nature of communication that was taking place. 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 MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques.
Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are

1 Encipherment
2 Digital Signature
3 Access Control

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.

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

Playfair cipher

The best known multiple letter encryption cipher is the playfair, which treats digrams in the plaintext as single units and translates these units into cipher text digrams. The playfair algorithm is based on the use of 5x5 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

Corresponding cipher text => CL KL CL RS PD IL HY AV MP HF XL IU

Strength of playfair cipher

Playfair cipher is a great advance over simple mono alphabetic ciphers.

Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual diagram is more difficult.

1.15.1.3 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

	PLAIN TEXT															
K		a	b	c	d	e	f	g	h	i	j	k	…	x	y	z
E	a	A	B	C	D	E	F	G	H	I	J	K	…	X	Y	Z
Y	b	B	C	D	E	F	G	H	I	J	K	L	…	Y	Z	A
	c	C	D	E	F	G	H	I	J	K	L	M	…	Z	A	B
L	d	D	E	F	G	H	I	J	K	L	M	N	…	A	B	C
E	e	E	F	G	H	I	J	K	L	M	N	O	…	B	C	D
T	f	F	G	H	I	J	K	L	M	N	O	P	…	C	D	E
T	g	G	H	I	J	K	L	M	N	O	P	Q	…	D	E	F
E	:	:	:	:	:	:	:	:	:	:	:	:	…	:	:	:
R	:	:	:	:	:	:	:	:	:	:	:	:		:	:	:
S	x	X	Y	Z	A	B	C	D	E	F	G	H	…			W
	y	Y	Z	A	B	C	D	E	F	G	H	I	…			X
	z	Z	A	B	C	D	E	F	G	H	I	J	…			Y

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 PT = 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 CT = ZICVTWQNGRZGVTWAVZHCQYGLMGJ

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

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

One Time Pad Cipher

It is an unbreakable cryptosystem. It represents the message as a sequence of 0s and 1s. This can be accomplished by writing all numbers in binary, for example, or by using ASCII. The key is a random sequence of 0‟s and 1‟s of same length as the message. Once a key is used, it is discarded and never used again. The system can be expressed as follows:

Ci = Pi Ki Ci - ith binary digit of cipher text Pi - ith binary digit of
plaintext Ki - ith binary digit of key

Exclusive OR operation
Thus the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key. Because of the properties of XOR, decryption simply involves the same bitwise operation:
Pi = Ci Ki
e.g., plaintext = 0 0 1 0 1 0 0 1
Key = 1 0 1 0 1 1 0 0
------------------ ciphertext = 1 0 0 0 0 1 0 1

Advantage:

- Encryption method is completely unbreakable for a ciphertext only attack.

Disadvantages

- It requires a very long key which is expensive to produce and expensive to transmit.
- Once a key is used, it is dangerous to reuse it for a second message; any knowledge on the first message would give knowledge of the second.

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.

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 L0 and R0. 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 Li-1 and Ri-1, derived from the previous round, as well as the subkey Ki, derived from the overall key K. in general, the subkeys Ki are different from K and from each other.

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 ki.

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.

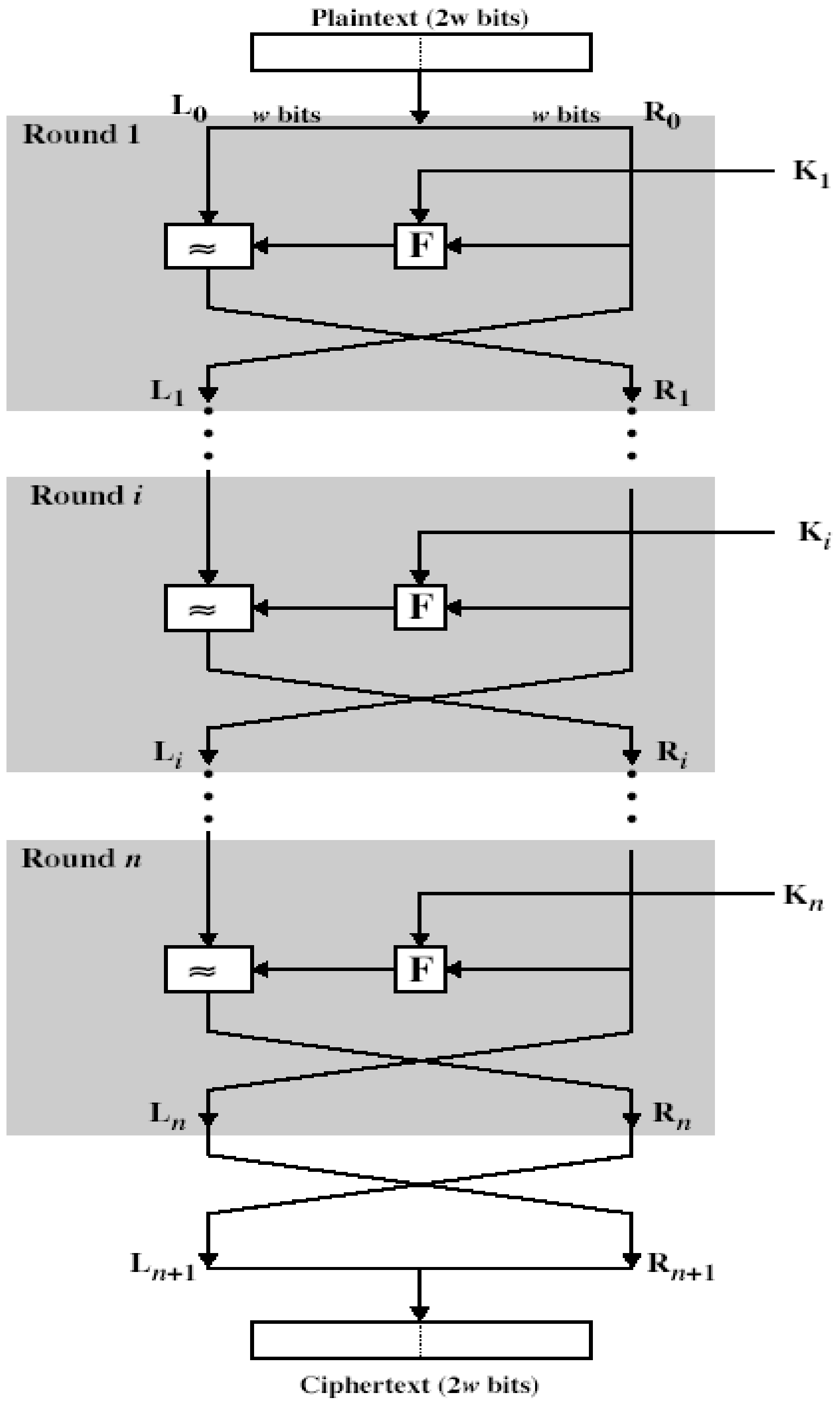


Fig: Classical Feistel Network

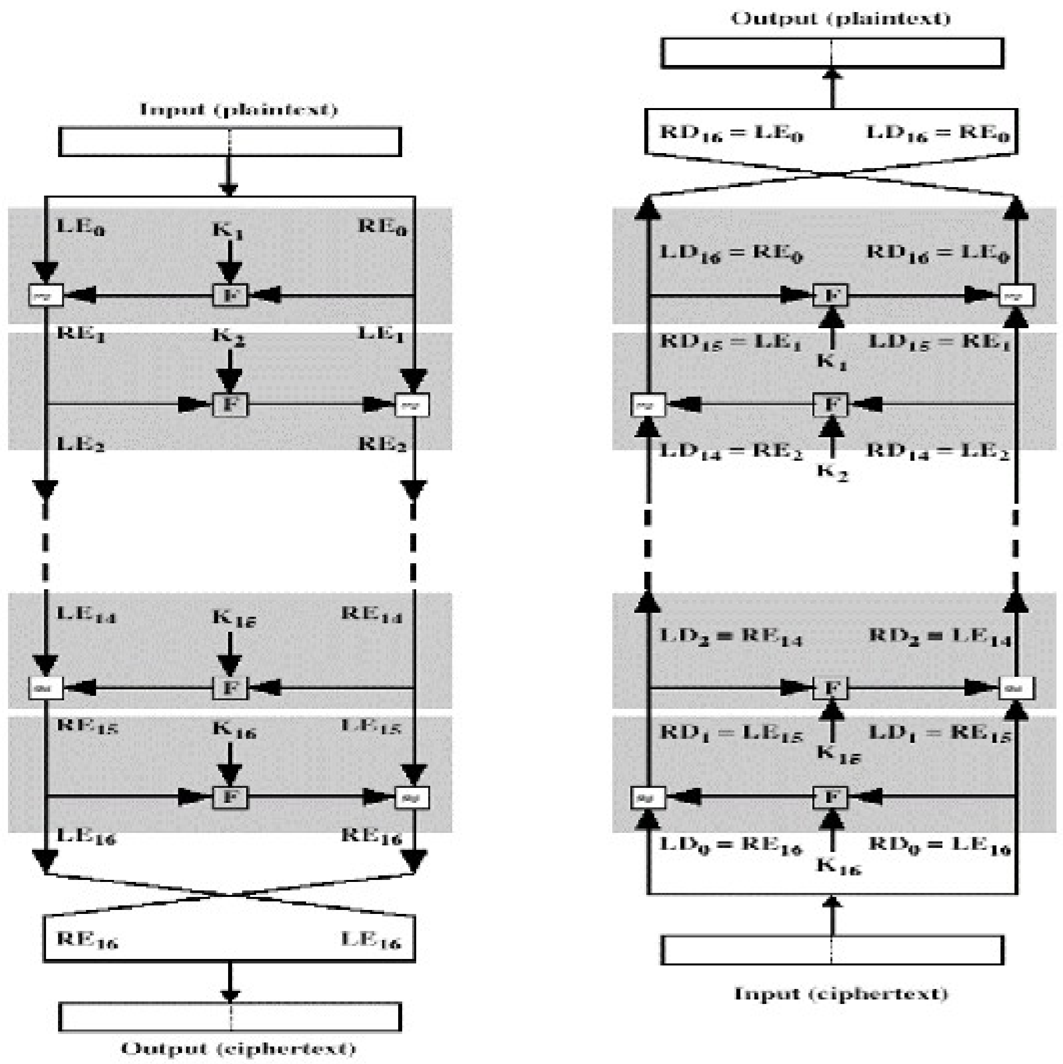


Fig: Feistel encryption and decryption

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 ki in reverse order. i.e., kn in the first round, kn-1 in second round and so on. For clarity, we use the notation LEi and REi 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., REi || LEi (or) equivalently RD16-i || LD16-i

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is RE16 || LE16. 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 RE16 || LE16, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.

Now we will see how the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process. First consider the encryption process,

LE16 = RE15
RE16 = LE15 F (RE15, K16) On the decryption side,
LD1 =RD0 = LE16 =RE15
RD1 = LD0 F (RD0, K16)
= RE16 F (RE15, K16)
= [LE15 F (RE15, K16)] F (RE15, K16)
= LE15
Therefore, LD1 = RE15
RD1 = LE15 In general, for the ith iteration of the encryption algorithm, LEi = REi-1
REi = LEi-1 F (REi-1, Ki)
Finally, the output of the last round of the decryption process is RE0 || LE0. A 32-bit swap recovers the original plaintext.

STEGANOGRAPHY

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.
e.g., (i) the sequence of first letters of each word of the overall message spells out the real (Hidden) message.
(ii) Subset of the words of the overall message is used to convey the hidden message.

Various other techniques have been used historically, some of them are

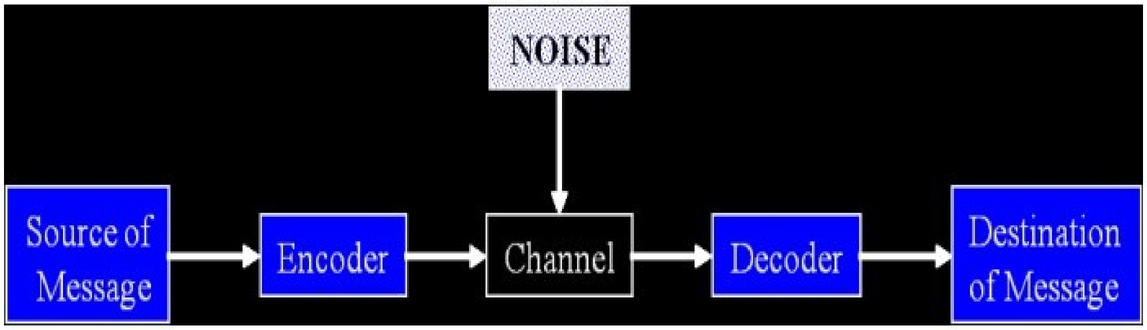
Character marking – selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held to an angle to bright light.
Invisible ink – a number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.
Pin punctures – small pin punctures on selected letters are ordinarily not visible unless the paper is held in front of the light. Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.
Typewriter correction ribbon: Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of steganography

- Requires a lot of overhead to hide a relatively few bits of information.
- Once the system is discovered, it becomes virtually worthless.

INFORMATION THEORY

Information theory is a branch of science that deals with the analysis of a communications system. We will study digital communications – using a file (or network protocol) as the channel Claude Shannon Published a landmark paper in 1948 that was the beginning of the branch of information theory We are interested in communicating information from a source to a destination In our case, the messages will be a sequence of binary digits Does anyone know the term for a binary digit.



One detail that makes communicating difficult is noise noise introduces uncertainty Suppose I wish to transmit one bit of information what are all of the possibilities tx 0, rx 0 - good tx 0, rx 1 - error tx 1, rx 0 - error tx 1, rx 1 - good Two of the cases above have errors – this is where probability fits into the picture In the case of steganography, the noise may be due to attacks on the hiding algorithm. Claude Shannon introduced the idea of self-information.

$$I(Xj) = \log\frac{1}{p(xj)} = \log\frac{1}{p(j)} = -\log pj$$

Suppose we have an event X, where Xi represents a particular outcome of the

Consider flipping a fair coin, there are two equiprobable outcomes: say X0 = heads, P0 = 1/2, X1 = tails, P1 = 1/2 The amount of self-information for any single result is 1 bit. In other words, the number of bits required to communicate the result of the event is 1 bit. When outcomes are equally likely, there is a lot of information in the result. The higher the likelihood of a particular outcome, the less information that outcome conveys However, if the coin is biased such that it lands with heads up 99% of the time, there is not much information conveyed when we flip the coin and it lands on heads. Suppose we have an event X, where Xi represents a particular outcome of the event. Consider flipping a coin, however, let‘s say there are 3 possible outcomes: heads (P = 0.49), tails (P=0.49), lands on its side (P = 0.02) – (likely much higher than in reality).

Information

There is no some exact definition, however:Information carries new specific knowledge, which is definitely new for its recipient; Information is always carried by some specific carrier in different forms (letters, digits, different specific symbols, sequences of digits, letters, and symbols , etc.); Information is meaningful only if the recipient is able to interpret it. According to the Oxford English Dictionary, the earliest historical meaning of the word information in English was the act of informing, or giving form or shape to the mind. The English word was apparently derived by adding the common "noun of action" ending "-action the information materialized is a message.

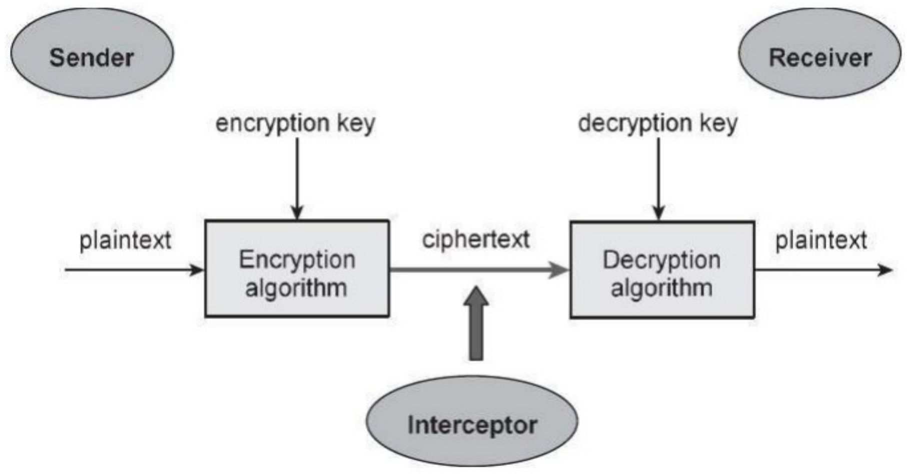
Information is always about something (size of a parameter, occurrence of an event, etc). Viewed in this manner, information does not have to be accurate; it may be a truth or a lie. Even a disruptive noise used to inhibit the flow of communication and create misunderstanding would in this view be a form of information. However, generally speaking, if the amount of information in the received message increases, the message is more accurate. Information Theory How we can measure the amount of information? How we can ensure the correctness of information? What to do if information gets corrupted by errors? How much memory does it require to store information? Basic answers to these questions that formed a solid background of the modern information theory were given by the great American mathematician, electrical engineer, and computer scientist Claude E. Shannon in his paper ―A Mathematical Theory of Communication‖ published in ―The Bell System Technical Journal in October, 1948.

A noiseless binary channel 0 0 transmits bits without error, What to do if we have a noisy channel and you want to send information across reliably? Information Capacity Theorem (Shannon Limit) The information capacity (or channel capacity) C of a continuous channel with bandwidth BHertz can be perturbed by additive Gaussian white noise of power spectral density N0/2, C=B log2(1+P/N0B) bits/sec provided bandwidth B satisfies where P is the average transmitted power P = Eb Rb (for an ideal system, Rb= C). Eb is the transmitted energy per bit, Rb is transmission rate.

CRYPTOSYSTEMS

A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system.

Let us discuss a simple model of a cryptosystem that provides confidentiality to the information being transmitted. This basic model is depicted in the illustration below −



The illustration shows a sender who wants to transfer some sensitive data to a receiver in such a way that any party intercepting or eavesdropping on the communication channel cannot extract the data. The objective of this simple cryptosystem is that at the end of the process, only the sender and the receiver will know the plaintext.

Components of a Cryptosystem

The various components of a basic cryptosystem are as follows −

Plaintext. It is the data to be protected during transmission.
Encryption Algorithm. It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
Ciphertext. It is the scrambled version of the plaintext produced by the encryption algorithm using a specific the encryption key. The ciphertext is not guarded. It flows on public channel. It can be intercepted or compromised by anyone who has access to the communication channel.
Decryption Algorithm. It is a mathematical process, that produces a unique plaintext for any given ciphertext and decryption key. It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.
Encryption Key. It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
Decryption Key. It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.

For a given cryptosystem, a collection of all possible decryption keys is called a **key space**.

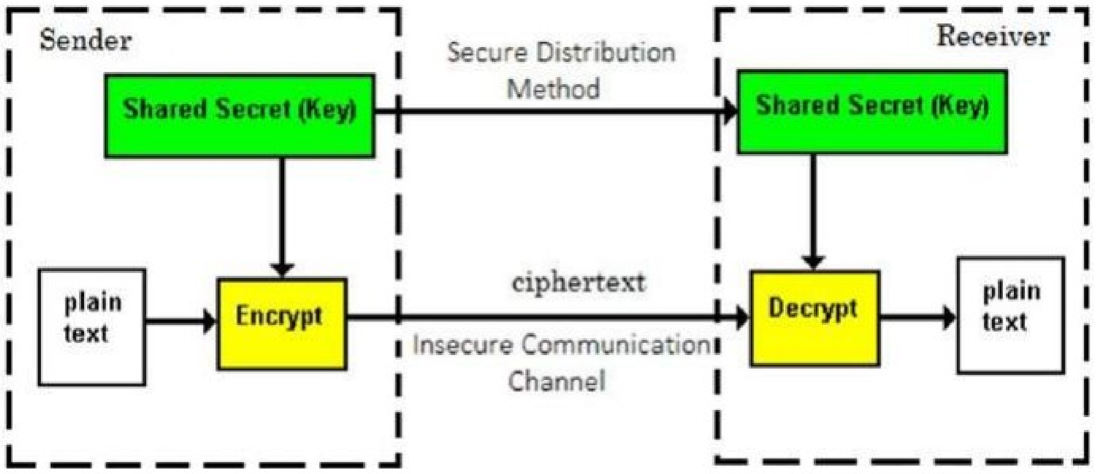
An **interceptor** anattacker is an unauthorized entity who attempts to determine the plaintext. He can see the ciphertext and may know the decryption algorithm. He, however, must never know the decryption key.

Types of Cryptosystems
Fundamentally, there are two types of cryptosystems based on the manner in which encryptiondecryption 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 × n– 1/2.

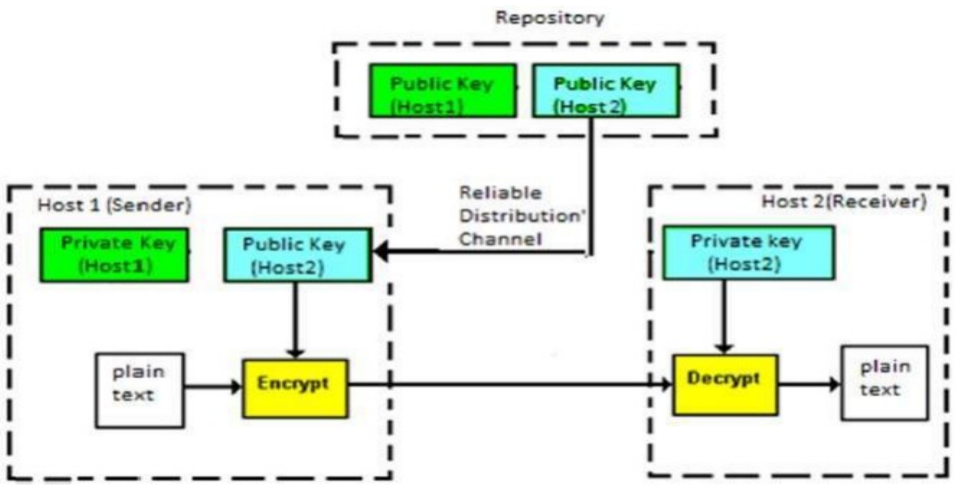
- Length of Key numberofbits in this encryption is smaller and hence, process of encryptiondecryption is faster than asymmetric key encryption.
- Processing power of computer system required to run symmetric algorithm is less.

Challenge of Symmetric Key Cryptosystem
There are two restrictive challenges of employing symmetric key cryptography.

Key establishment − Before any communication, both the sender and the receiver need to agree on a secret symmetric key. It requires a secure key establishment mechanism in place.
Trust Issue − Since the sender and the receiver use the same symmetric key, there is an implicit requirement that the sender and the receiver 'trust' each other. For example, it may happen that the receiver has lost the key to an attacker and the sender is not informed.

These two challenges are highly restraining for modern day communication. Today, people need to exchange information with non-familiar and non-trusted parties. For example, a communication between online seller and customer. These limitations of symmetric key encryption gave rise to asymmetric key encryption schemes.

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 20 th century to come over the necessity of preshared 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 numberofbits in this encryption is large and hence, the process of encryptiondecryption is slower than symmetric key encryption.
- Processing power of computer system required to run asymmetric algorithm is higher.

Symmetric cryptosystems are a natural concept. In contrast, public-key cryptosystems are quite difficult to comprehend.
You may think, how can the encryption key and the decryption key are 'related', and yet it is impossible to determine the decryption key from the encryption key? The answer lies in the mathematical concepts. It is possible to design a cryptosystem whose keys have this property. The concept of public-key cryptography is relatively new. There are fewer public-key algorithms known than symmetric algorithms.

Challenge of Public Key Cryptosystem
Public-key cryptosystems have one significant challenge − the user needs to trust that the public key that he is using in communications with a person really is the public key of that person and has not been spoofed by a malicious third party.
This is usually accomplished through a Public Key Infrastructure PKI consisting a trusted third party. The third party securely manages and attests to the authenticity of public keys. When the third party is requested to provide the public key for any communicating person X, they are trusted to provide the correct public key.
The third party satisfies itself about user identity by the process of attestation, notarization, or some other process − that X is the one and only, or globally unique, X. The most common method of making the verified public keys available is to embed them in a certificate which is digitally signed by the trusted third party.

Relation between Encryption Schemes
A summary of basic key properties of two types of cryptosystems is given below −

	Symmetric Cryptosystems	**Public Key Cryptosystems**
Relation between Keys	Same	Different, but mathematically related
Encryption Key	Symmetric	Public
Decryption Key	Symmetric	Private

Due to the advantages and disadvantage of both the systems, symmetric key and public-key cryptosystems are often used together in the practical information security systems.

Kerckhoff's Principle for Cryptosystem
In the 19 th century, a Dutch cryptographer A. Kerckhoff furnished the requirements of a good cryptosystem. Kerckhoff stated that a cryptographic system should be secure even if everything about the system, except the key, is public knowledge. The six design principles defined by Kerckhoff for cryptosystem are −

- The cryptosystem should be unbreakable practically, if not mathematically.
- Falling of the cryptosystem in the hands of an intruder should not lead to any compromise of the system, preventing any inconvenience to the user.
- The key should be easily communicable, memorable, and changeable.
- The ciphertext should be transmissible by telegraph, an unsecure channel.
- The encryption apparatus and documents should be portable and operable by a single person.
- Finally, it is necessary that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

The second rule is currently known as Kerckhoff principle. It is applied in virtually all the contemporary encryption algorithms such as DES, AES, etc. These public algorithms are considered to be thoroughly secure. The security of the encrypted message depends solely on the security of the secret encryption key.
Keeping the algorithms secret may act as a significant barrier to cryptanalysis. However, keeping the algorithms secret is possible only when they are used in a strictly limited circle.
In modern era, cryptography needs to cater to users who are connected to the Internet. In such cases, using a secret algorithm is not feasible, hence Kerckhoff principles became essential guidelines for designing algorithms in modern cryptography.

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.
Cipher text only – A copy of cipher text alone is known to the cryptanalyst.
Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.
Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.
Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

UNIT - II

Symmetric Block Encryption Algorithms

Data Encryption Standard (DES): An encryption algorithm that encrypts data with a 56-bit, randomly generated symmetric key. DES is not a secure encryption algorithm and it was cracked many times. Data Encryption Standard (DES) was developed by IBM and the U.S. Government together. DES is a block encryption algorithm.

Data Encryption Standard XORed (DESX): DESX is a stronger variation of the DES encryption algorithm. In DESX, the input plaintext is bitwise XORed with 64 bits of additional key material before encryption with DES and the output is also bitwise XORed with another 64 bits of key material.

Triple DES (3DES): Triple DES was developed from DES, uses a 64-bit key consisting of 56 effective key bits and 8 parity bits. In 3DES, DES encryption is applied three times to the plaintext. The plaintext is encrypted with key A, decrypted with key B, and encrypted again with key C. 3DES is a block encryption algorithm.

RC2 and RC5: Ronald Rivest (RSA Labs), developed these algorithms. They are block encryption algorithms with variable block and key sizes. It is difficult to break if the attacker does not know the original sizes when attempting to decrypt captured data.

RC4: A variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation and is commonly used for the encryption of traffic to and from secure Web sites using the SSL protocol.

Advanced Encryption Standard (AES): Advanced Encryption Standard (AES) is a newer and stronger encryption standard, which uses the Rijndael (pronounced Rhine-doll) algorithm. This algorithm was developed by Joan Daemen and Vincent Rijmen of Belgium. AES will eventually displace DESX and 3DES. AES is capable to use 128-bit, 192-bit, and 256-bit keys.

International Data Encryption Algorithm (IDEA): IDEA encryption algorithm is the European counterpart to the DES encryption algorithm. IDEA is a block cipher, designed by Dr. X. Lai and Professor J. Massey. It operates on a 64-bit plaintext block and uses a 128-bit key. IDEA uses a total of eight rounds in which it XOR's, adds and multiplies four sub-blocks with each other, as well as six 16-bit sub-blocks of key material.

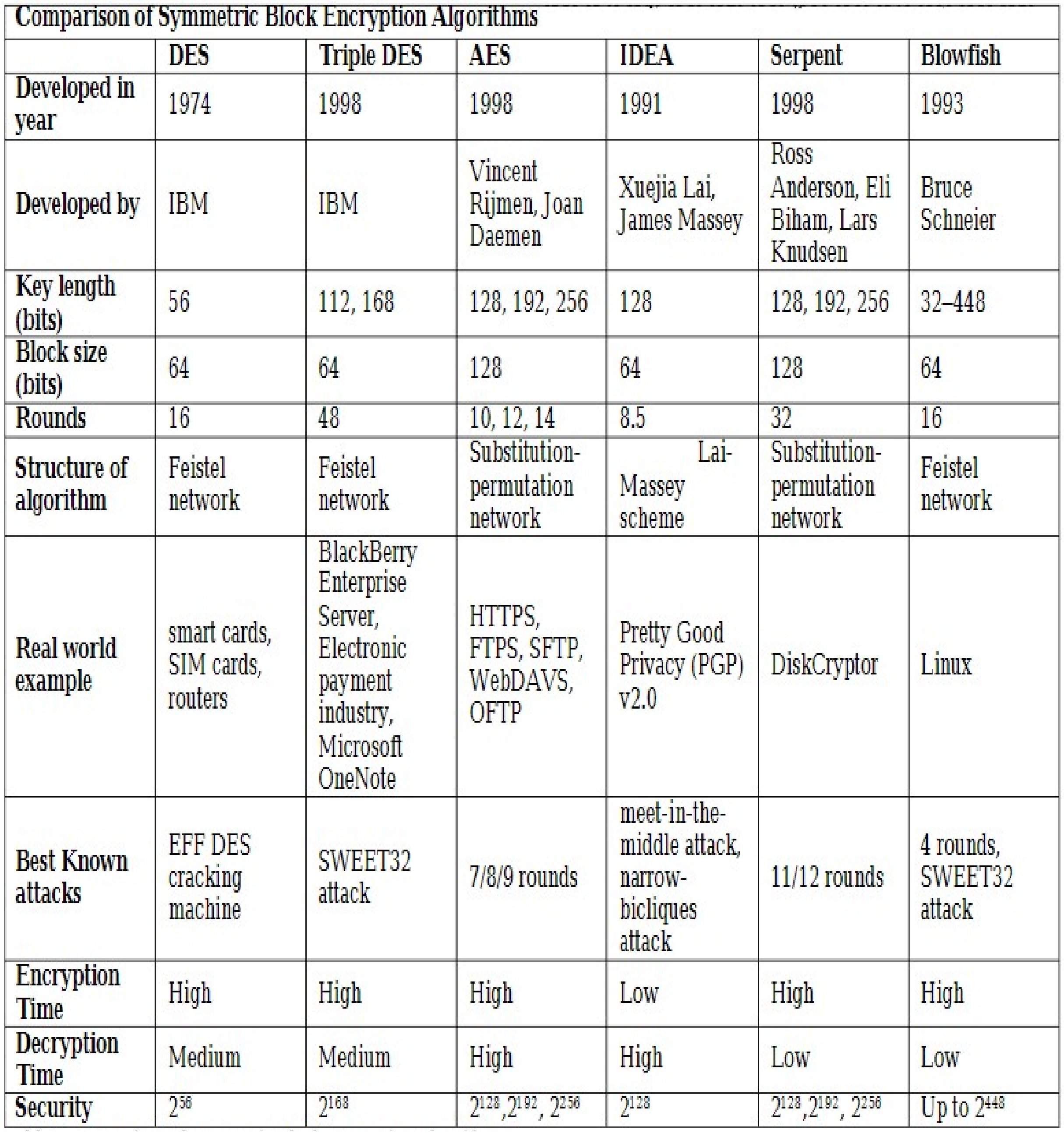
Blowfish: Blowfish is a symmetric block cipher, designed by Bruce Schneier. Blowfish has a 64-bit block size and a variable key length from 32 up to 448 bits. Bruce Schneier later created Twofish, which performs a similar function on 128-bit blocks.

CAST: CAST is an algorithm developed by Carlisle Adams and Stafford Tavares. It's used in some products offered by Microsoft and IBM. CAST uses a 40-bit to 128-bit key, and it's very fast and efficient.

Note:

Block Cipher: A block cipher divides data into chunks, pads the last chunk if necessary, and then encrypts each chunk in its turn.

Streaming Cipher. A streaming cipher uses a series of random numbers seeded with a cipher key to encrypt a stream of bits.

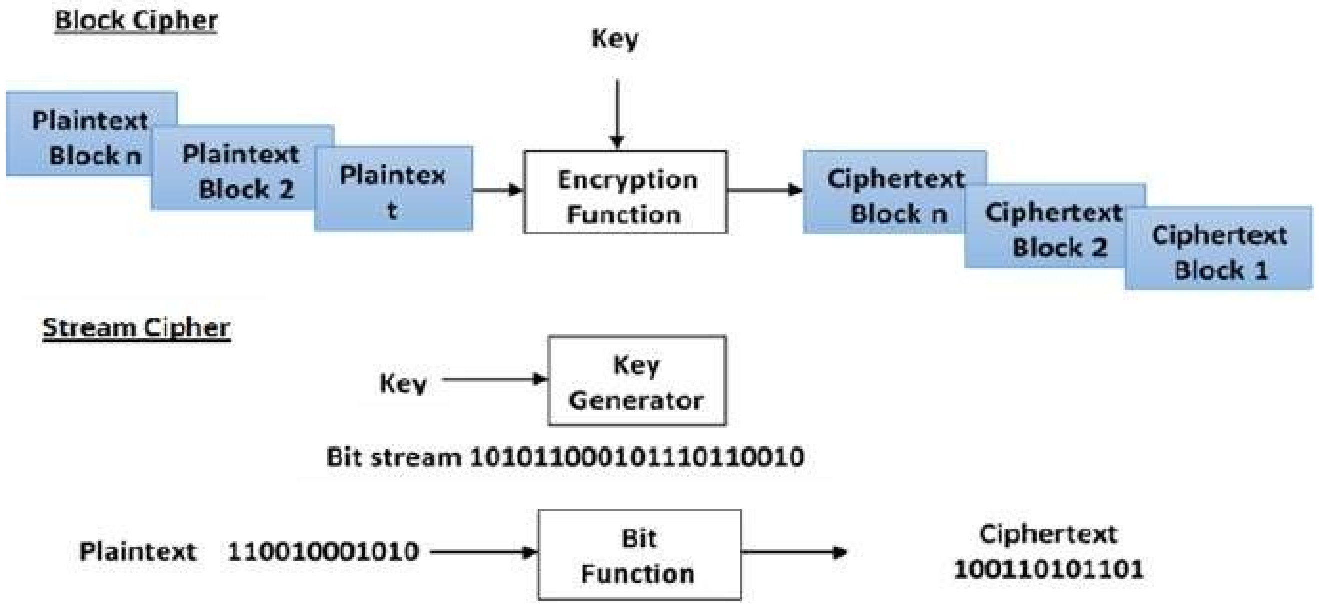


Comparison of Symmetric Block Encryption Algorithms

	DES	**Triple DES**	**AES**	**IDEA**	**Serpent**	**Blowfish**
Developed in year	1974	1998	1998	1991	1998	1993
Developed by	IBM	IBM	Vincent Rijmen, Joan Daemen	Xuejia Lai, James Massey	Ross Anderson, Eli Biham, Lars Knudsen	Bruce Schneier
Key length (bits)	56	112, 168	128, 192, 256	128	128, 192, 256	32–448
Block size (bits)	64	64	128	64	128	64
Rounds	16	48	10, 12, 14	8.5	32	16
Structure of algorithm	Feistel network	Feistel network	Substitution-permutation network	Lai-Massey scheme	Substitution-permutation network	Feistel network
Real world example	smart cards, SIM cards, routers	BlackBerry Enterprise Server, Electronic payment industry, Microsoft OneNote	HTTPS, FTPS, SFTP, WebDAVS, OFTP	Pretty Good Privacy (PGP) v2.0	DiskCryptor	Linux
Best Known attacks	EFF DES cracking machine	SWEET32 attack	7/8/9 rounds	meet-in-the-middle attack, narrow-bicliques attack	11/12 rounds	4 rounds, SWEET32 attack
Encryption Time	High	High	High	Low	High	High
Decryption Time	Medium	Medium	High	High	Low	Low
Security	2^{56}	2^{168}	$2^{128}, 2^{192}, 2^{256}$	2^{128}	$2^{128}, 2^{192}, 2^{256}$	Up to 2^{448}

Stream Ciphers

In this scheme, the plaintext is processed one bit at a time i.e. one bit of plaintext is taken, and a series of operations is performed on it to generate one bit of ciphertext. Technically, stream ciphers are block ciphers with a block size of one bit.



RC4 Encryption Algorithm

RC4 is a stream cipher and variable length key algorithm. This algorithm encrypts one byte at a time (or larger units on a time).

A key input is pseudorandom bit generator that produces a stream 8-bit number that is unpredictable without knowledge of input key, The output of the generator is called key-stream, is combined one byte at a time with the plaintext stream cipher using X-OR operation.

Example:

RC4 Encryption

10011000 ? 01010000 = 11001000

RC4 Decryption

11001000 ? 01010000 = 10011000

Key-Generation Algorithm –

A variable-length key from 1 to 256 byte is used to initialize a 256-byte state vector S, with elements S[0] to S[255]. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion, then the entries in S are permuted again.

1. **Key-Scheduling Algorithm:**

   Initialization: The entries of S are set equal to the values from 0 to 255 in ascending order, a temporary vector T, is created.

   If the length of the key k is 256 bytes, then k is assigned to T. Otherwise, for a key with length(k_len) bytes, the first k_len elements of T as copied from K and then K is repeated as many times as necessary to fill T. The idea is illustrated as follow:

```
for
   i = 0 to 255 do S[i] = i;
T[i] = K[i mod k_len];
```

   we use T to produce the initial permutation of S. Starting with S[0] to S[255], and for each S[i] algorithm swap it with another byte in S according to a scheme dictated by T[i], but S will still contain values from 0 to 255 :

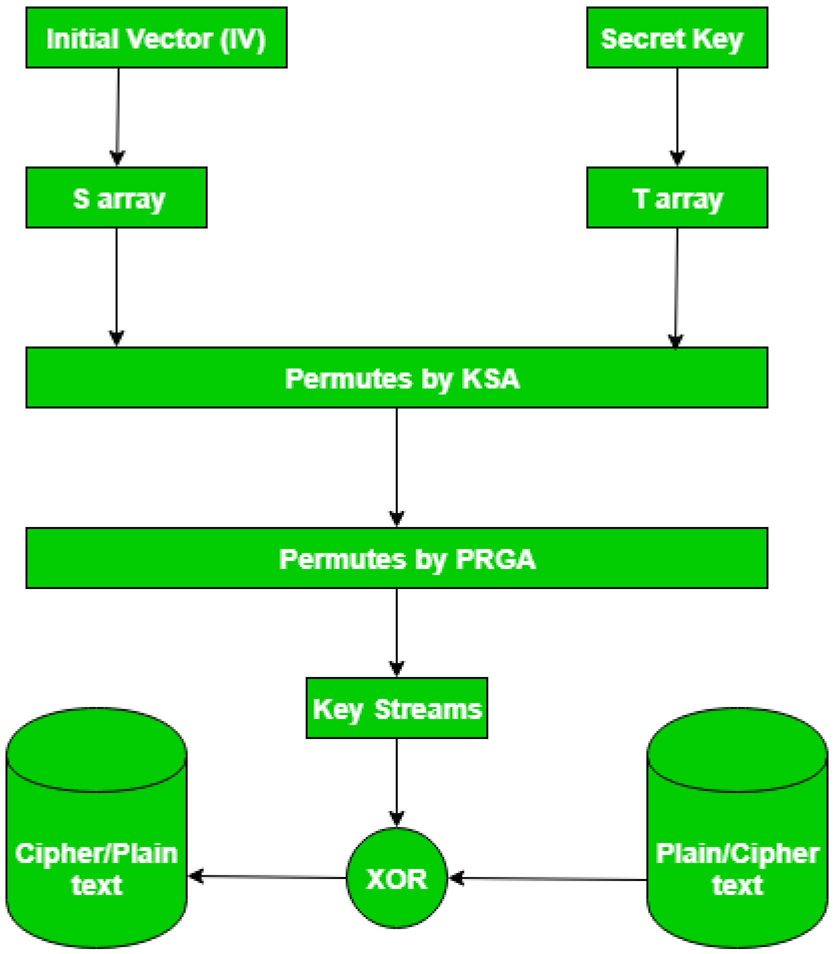
```
j = 0;
for
   i = 0 to 255 do
   {
      j = (j + S[i] + T[i])mod 256;
      Swap(S[i], S[j]);
   }
```

2. **Pseudo random generation algorithm (Stream Generation):**

   Once the vector S is initialized, the input key will not be used. In this step, for each S[i] algorithm swap it with another byte in S according to a scheme dictated by the current configuration of S. After reaching S[255] the process continues, starting from S[0] again

```
i, j = 0;
while (true)
   i = (i + 1)mod 256;
j = (j + S[i])mod 256;
Swap(S[i], S[j]);
t = (S[i] + S[j])mod 256;
k = S[t];
```

3. **Encrypt using X-Or():**



Block Cipher Modes of Operation

Here, we discuss the different modes of operation of a block cipher. These are procedural rules for a generic block cipher. Interestingly, the different modes result in different properties being achieved which add to the security of the underlying block cipher.

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.

Electronic Code Book (ECB) Mode

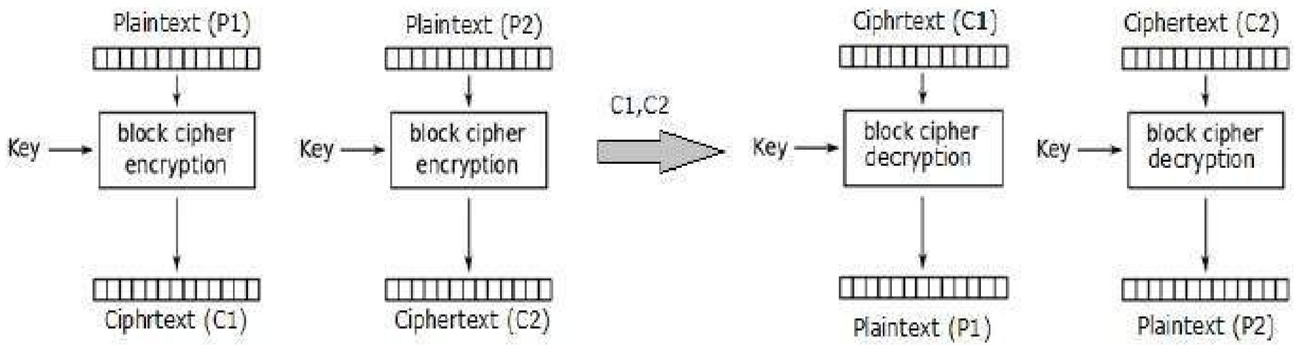
This mode is a most straightforward way of processing a series of sequentially listed message blocks.

Operation

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

The ECB mode is **deterministic**, that is, if plaintext block P1, P2,…, Pm are encrypted twice under the same key, the output ciphertext blocks will be the same.

In fact, for a given key technically we can create a codebook of ciphertexts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding ciphertext. Thus, the operation is analogous to the assignment of code words in a codebook, and hence gets an official name − Electronic Codebook mode of operation (ECB). It is illustrated as follows −



Analysis of ECB Mode

In reality, any application data usually have partial information which can be guessed. For example, the range of salary can be guessed. A ciphertext from ECB can allow an attacker to guess the plaintext by trial-and-error if the plaintext message is within predictable.

For example, if a ciphertext from the ECB mode is known to encrypt a salary figure, then a small number of trials will allow an attacker to recover the figure. In general, we do not wish to use a deterministic cipher, and hence the ECB mode should not be used in most applications.

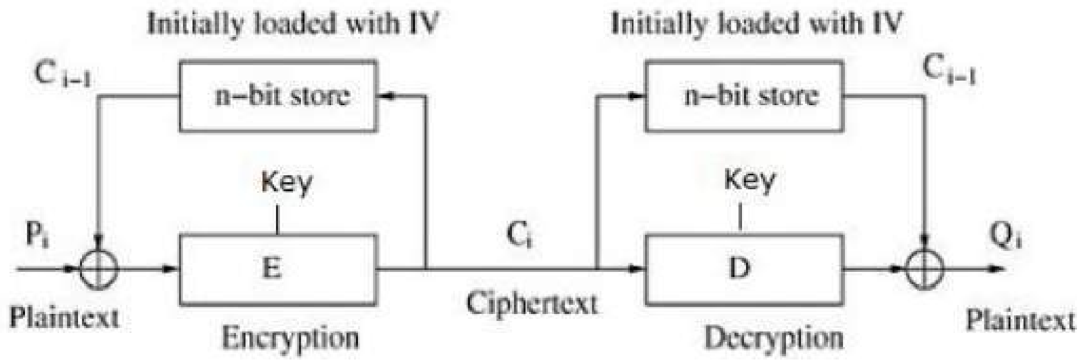
Cipher Block Chaining (CBC) Mode

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.

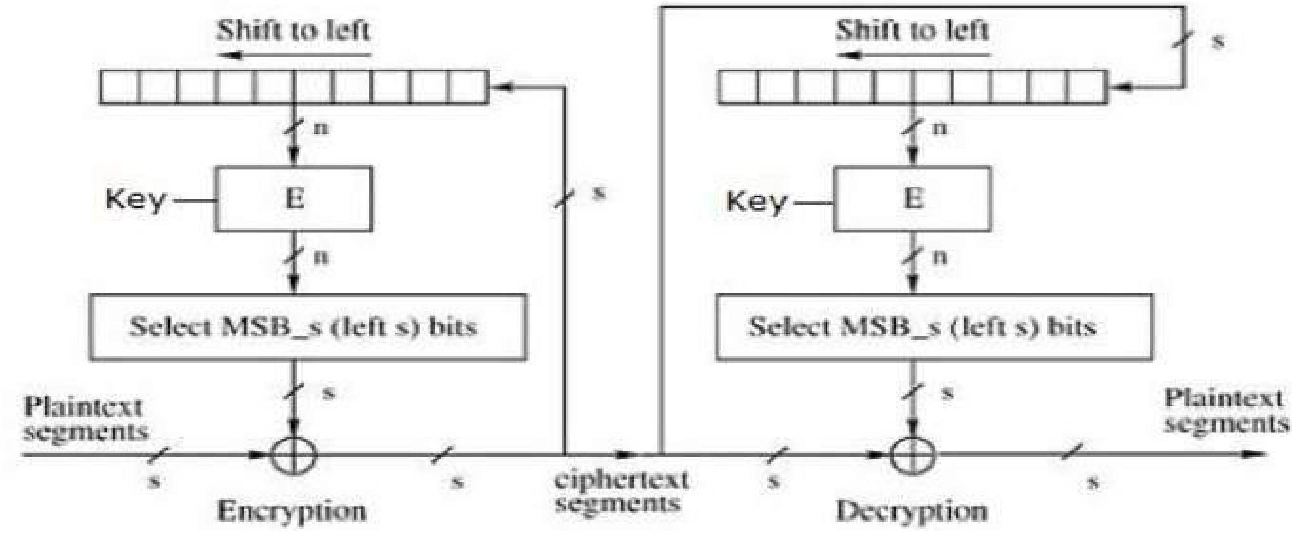
Cipher Feedback (CFB) Mode

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.

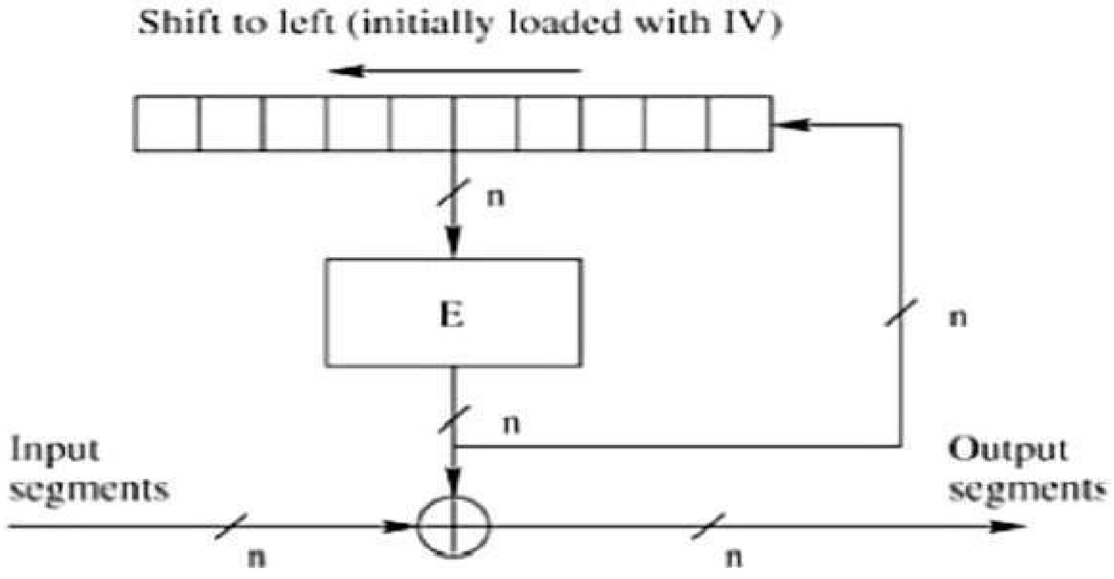
On the flip side, the error of transmission gets propagated due to changing of blocks.

Output Feedback (OFB) Mode

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

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

The operation is depicted in the following illustration −



Counter (CTR) Mode

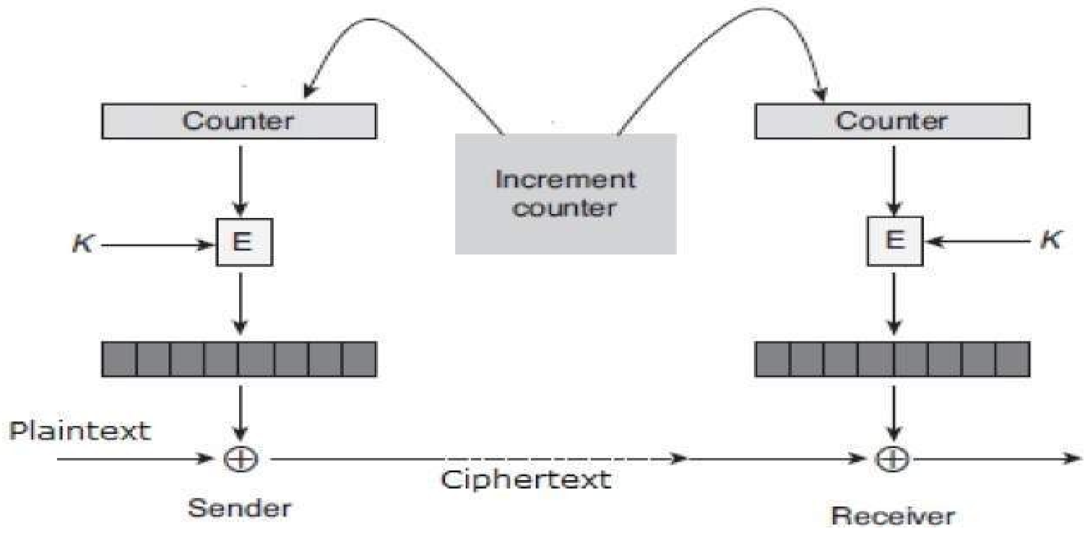
It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge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

Operation

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

- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.

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



Analysis of Counter Mode

It does not have message dependency and hence a ciphertext block does not depend on the previous plaintext blocks.

Like CFB mode, CTR mode does not involve the decryption process of the block cipher. This is because the CTR mode is really using the block cipher to generate a key-stream, which is encrypted using the XOR function. In other words, CTR mode also converts a block cipher to a stream cipher.

The serious disadvantage of CTR mode is that it requires a synchronous counter at sender and receiver. Loss of synchronization leads to incorrect recovery of plaintext.

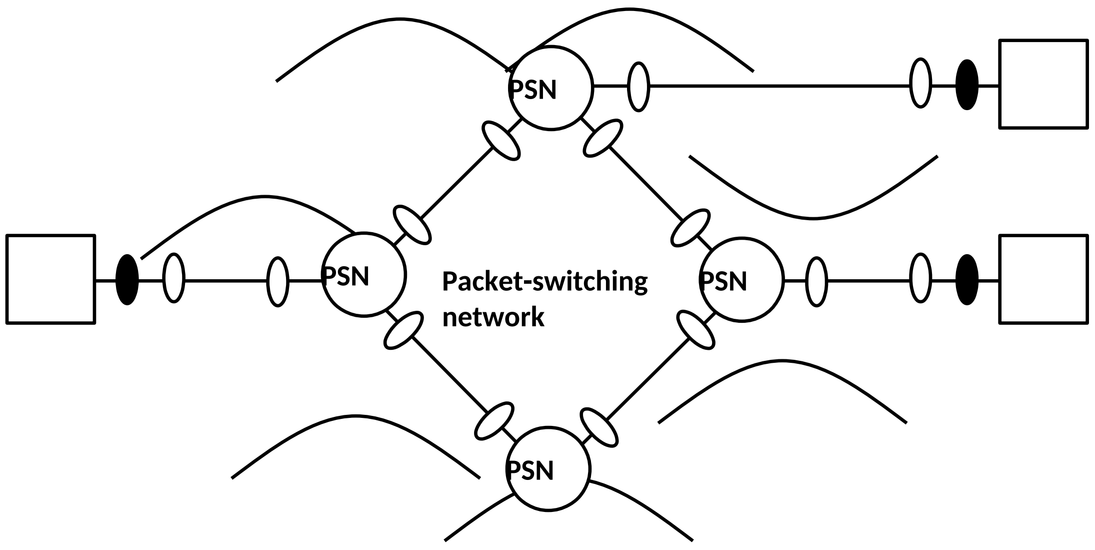
However, CTR mode has almost all advantages of CFB mode. In addition, it does not propagate error of transmission at all.

Location of Encryption Devices

As you should know by now, the most powerful and common approach to countering the threats to network security is encryption. In using encryption, we need to decide what to encrypt and where the encryption devices should be located. There are two fundamental alternatives:

- Link encryption
- End –to-end encryption

There are shown in use over a packet – switching network in Fig. 3.



● =End-to-end encryption Device

⬯ =Link encryption device

PSN =Packet switching node

Figure 3 Encryption across a Packet-Switching Network

Link Encryption

In this scheme, each vulnerable communications link is equipped on both ends with an encryption device. Thus all traffic over all communications links is secured. Although, this requires a lot of encryption devices in a large network, it provides a high level of security. One disadvantage of this approach is that the message must be decrypted each time it enters a packet switch. This is necessary because the packet switch must read the address (i.e., the virtual circuit number) in the packet header to route the packet. Thus the message is vulnerable at each switch. If this is a public packet-switching network (PSN), the user has no control over the security of the modes.

End-to-End Encryption

In this approach, 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 on host. The destination shares a key with the source and so is able to decrypt the data. This approach would seem to secure the transmission against attacks on the network links on switches. However, the host may only encrypt the used data position of the packet and must leave the header in the clear, so that it can be read by the network.

Note 1 – With end-to-end encryption, the user data are secure. However, the traffic pattern is not, because packet headers are transmitted in the.

Note 2 – To achieve greater security, both link and end-to-end encryptions are needed.

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:

1. A key could be selected by Alice and physically delivered to Bob:

2. A 3rd party could select the key and physically deliver it to Alice and Bob.

3. 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.

4. 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 3 – 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 4 – Option 3 is a possibility for either link encryption on end-to-end encryption, but it an opponent, Oscar, ever succeeds in gaining access to one key, then all subsequent keys are revealed.

Note 5 – To provide keys for end-to-end encryption, option 4 is preferable.
Key distribution Center(KDC)

Figure 4 shows an implementation that uses option 4 for end-to-end encryption. The trusted third party (TTP) becomes known as a key distribution center (KDC). The configuration consists of the following elements:

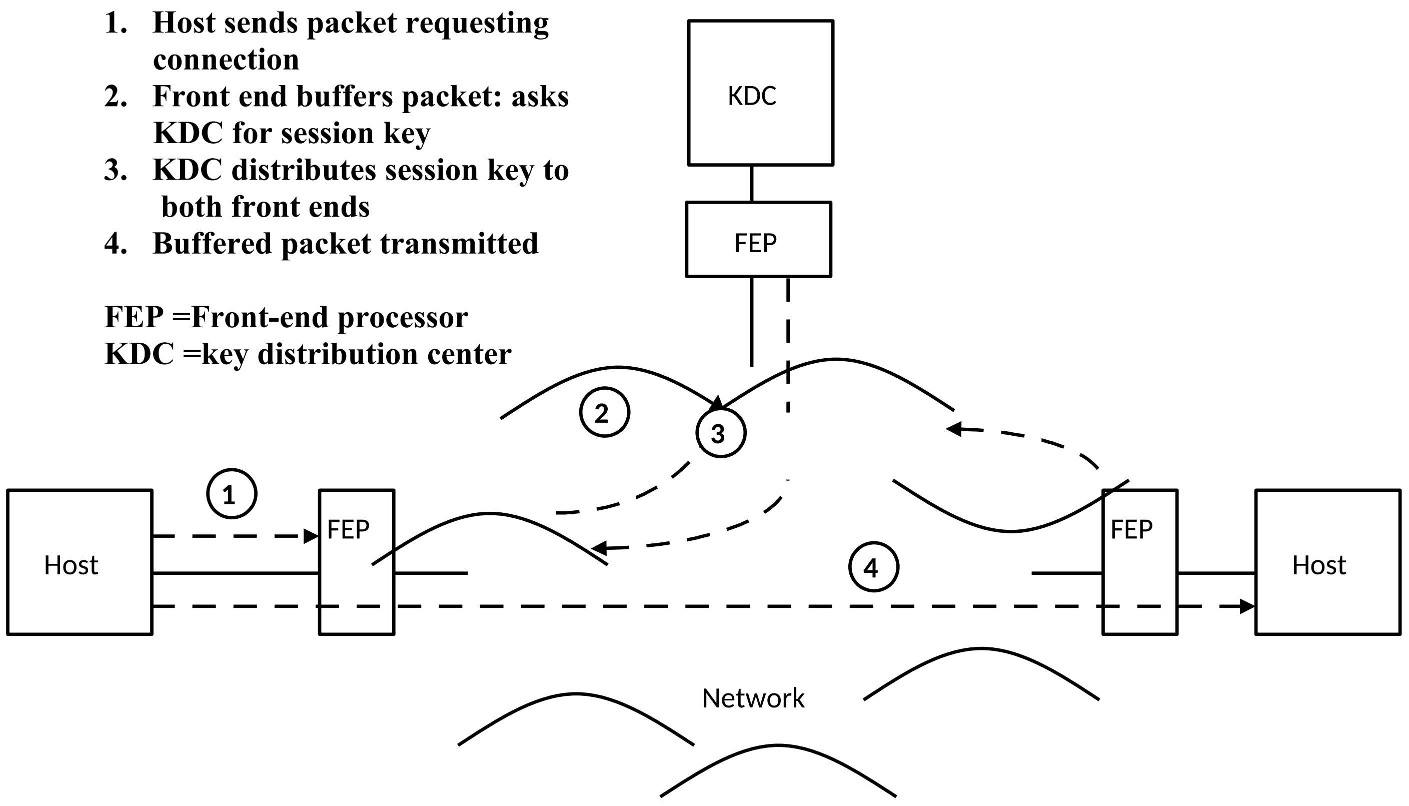

1. Host sends packet requesting connection
2. Front end buffers packet: asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted
FEP =Front-end processor
KDC =key distribution center
KDC
FEP
2
3
1
Host
FEP
4
FEP
Host
Network

Figure 4 Automatic Key Distribution for Connection-Oriented Protocol

1. KDC – Also known as key exchange authority or key exchange center, determine which systems are allowed to communicate with each other securely. When permission is granted, the KDC provides a one-time session key for that connection. The session keys are used for the duration of a session. At the conclusion of the session, or connection, the session key is destroyed.

2. FEP – The front-end procession (FEP) performs end-to-end encryption and obtains session keys on behalf of its host on terminal.

The steps involved in establishing a connection are shown in Fig. 4.

Step 1 – When one host wishes to set up a connection to another host, it transmit a connection sequent packet.

Step 2 – The FEP saves that packet and applies to the KDC for permission to establish the connection. The communications between the FEP and the KDC is encrypted using a master key shared only by the FEP and the KDC.

Step 3 – If the KDC approves the connection sequent, it generates the session key and delivers it to the two appropriate FEPs, using a unique permanent key for each FEP.

Step 4 – The requesting FEP can now release the connection sequent packet, and a connection is set up between the two and systems. All used data exchanged between the two end systems are encrypted by their respective FEPs using the one-time session key.

Note 6 – The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other. Kerberos, used extensively in Microsoft Windows 2000, is modeled on a KDC.

Note 7 – In general, a KDC supporting n sites, where each site needs a secret key with every other site, must make almost $n^2/2$ keys. This means that a KDC supporting 1,000 sites must make almost 500,000 keys, an unmanageable number of keys.

Note 8 – The KDC is often burdened with extensive key management and can become a bottleneck. Additionally, if the KDC also acts as a key escrow agent, the KDC itself is an attractive target (e.g., for a distributed denial-of-service attack). For these reasons, the symmetrical encryption should be avoided altogether. Another approach to security is the public-key encryption, which makes key distribution much easier.

Message Authentication

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

A message, file, document, or other collection of data is said to be authentic when it is genuine and came from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received message is authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.

1. Authentication Using Conventional Encryption

It is possible to perform authentication simply by the use of conventional encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

2. Message Authentication without Message Encryption

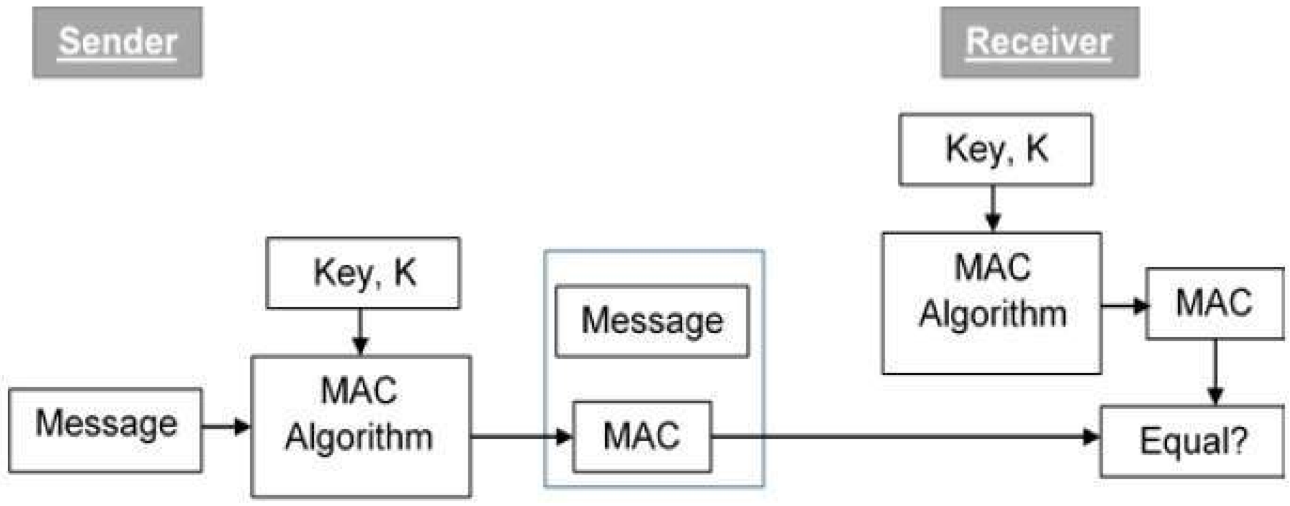
We examine several approaches to message authentication that do not rely on encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

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.

Both these limitations can be overcome by using the public key based digital signatures discussed in following section.

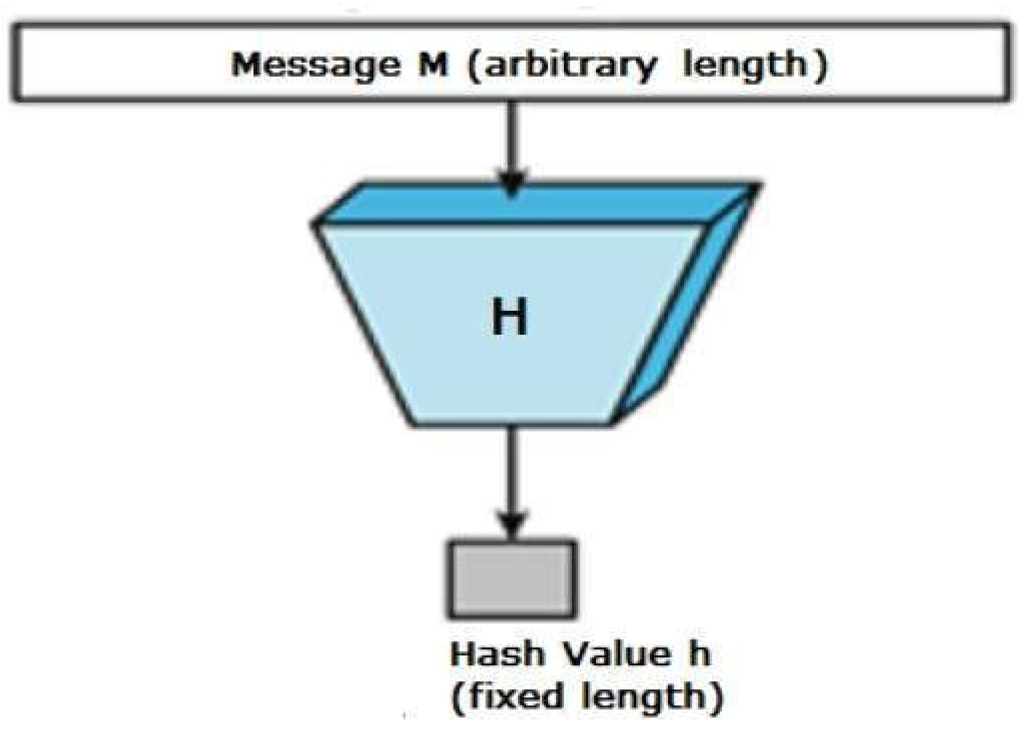
Secure Hash Functions and HMAC

Hash Functions

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 coverts 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.

Properties of Hash Functions

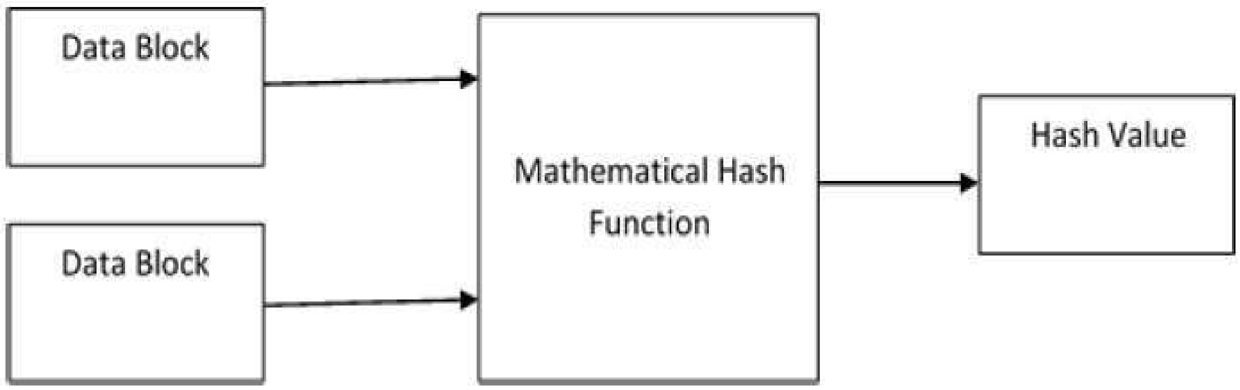
In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

- **Pre-Image Resistance**
  - This property means that it should be computationally hard to reverse a hash function.
  - In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
  - This property protects against an attacker who only has a hash value and is trying to find the input.
- **Second Pre-Image Resistance**
  - This property means given an input and its hash, it should be hard to find a different input with the same hash.
  - In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.
- **Collision Resistance**
  - This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
  - In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.
  - Also, if a hash function is collision-resistant **then it is second pre-image resistant.**

Design of Hashing Algorithms

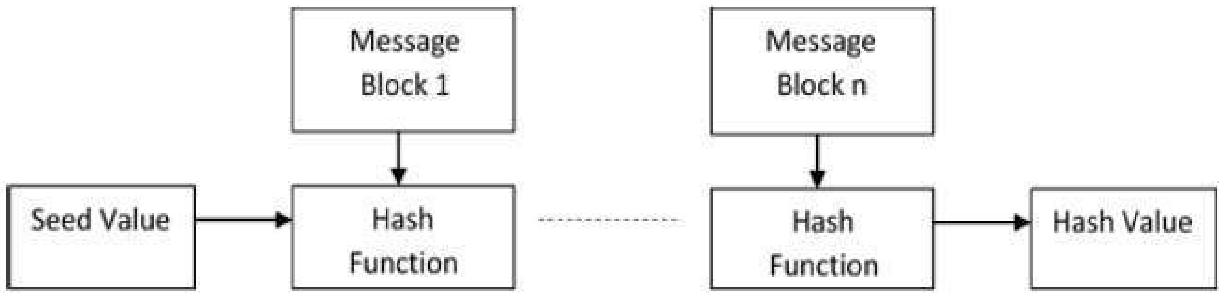
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

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 functions, specifying how the message will be broken up and how the results from previous message blocks are chained together.

Popular Hash Functions

Let us briefly see some popular hash functions −

Message Digest (MD)

MD5 was most popular and widely used hash function for quite some years.

- The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.

- MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it.
- In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

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.

RIPEMD

The RIPEND is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by open research community and generally known as a family of European hash functions.

- The set includes RIPEND, RIPEMD-128, and RIPEMD-160. There also exist 256, and 320-bit versions of this algorithm.
- Original RIPEMD (128 bit) is based upon the design principles used in MD4 and found to provide questionable security. RIPEMD 128-bit version came as a quick fix replacement to overcome vulnerabilities on the original RIPEMD.

- RIPEMD-160 is an improved version and the most widely used version in the family. The 256 and 320-bit versions reduce the chance of accidental collision, but do not have higher levels of security as compared to RIPEMD-128 and RIPEMD-160 respectively.

Whirlpool

This is a 512-bit hash function.

- It is derived from the modified version of Advanced Encryption Standard (AES). One of the designer was Vincent Rijmen, a co-creator of the AES.
- Three versions of Whirlpool have been released; namely WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL.

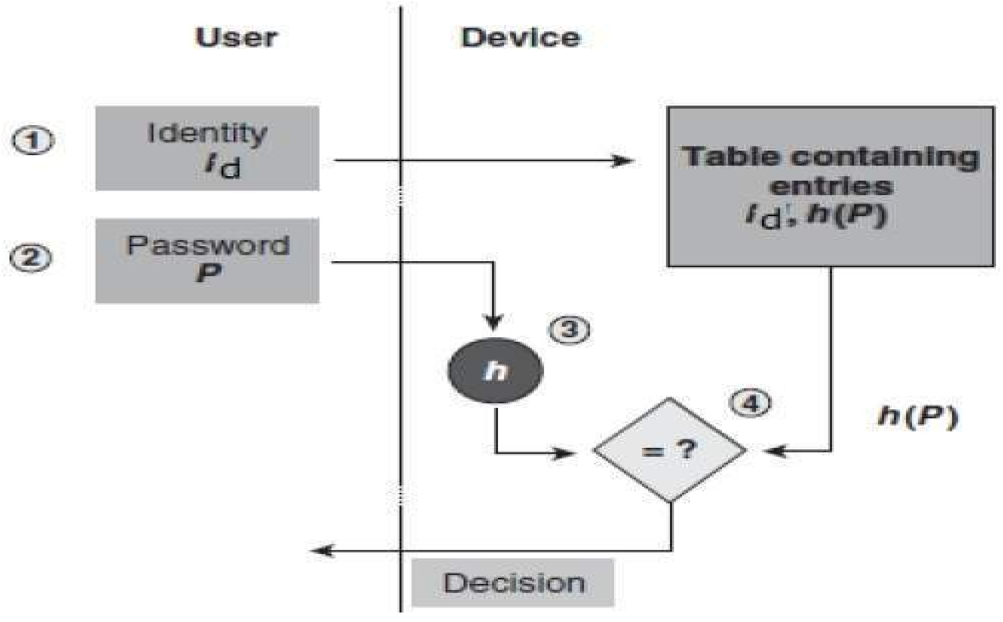
Applications of Hash Functions

There are two direct applications of hash function based on its cryptographic properties.

Password Storage

Hash functions provide protection to password storage.

- Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.
- The Password file consists of a table of pairs which are in the form (user id, h(P)).
- The process of logon is depicted in the following illustration −

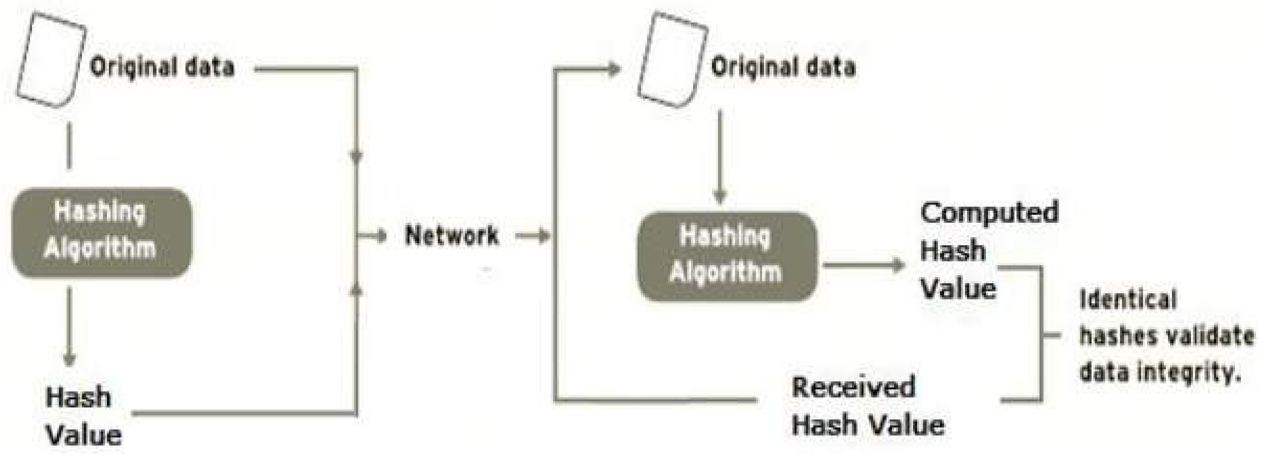


- An intruder can only see the hashes of passwords, even if he accessed the password. He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

Data Integrity Check

Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files. This application provides assurance to the user about correctness of the data.

The process is depicted in the following illustration −

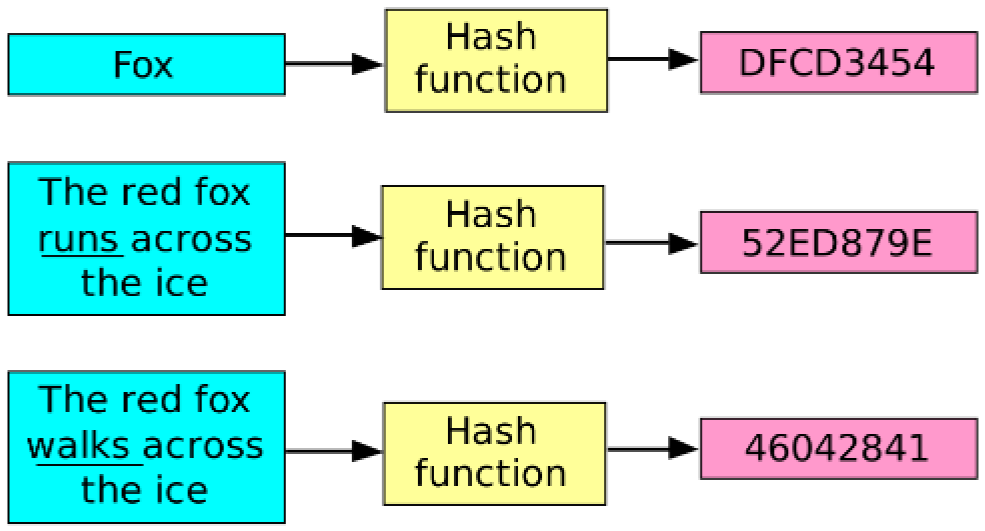


The integrity check helps the user to detect any changes made to original file. It however, does not provide any assurance about originality. The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver. This integrity check application is useful only if the user is sure about the originality of file.

Secure Hash Algorithms

Secure Hash Algorithms, also known as SHA, are a family of cryptographic functions designed to keep data secured. It works by transforming the data using a hash function: an algorithm that consists of bitwise operations, modular additions, and compression functions. The hash function then produces a fixed-size string that looks nothing like the original. These algorithms are designed to be one-way functions, meaning that once they're transformed into their respective hash values, it's virtually impossible to transform them back into the original data. A few algorithms of interest are SHA-1, SHA-2, and SHA-3, each of which was successively designed with increasingly stronger encryption in response to hacker attacks. SHA-0, for instance, is now obsolete due to the widely exposed vulnerabilities.

A common application of SHA is to encrypting passwords, as the server side only needs to keep track of a specific user's hash value, rather than the actual password. This is helpful in case an attacker hacks the database, as they will only find the hashed functions and not the actual passwords, so if they were to input the hashed value as a password, the hash function will convert it into another string and subsequently deny access. Additionally, SHAs exhibit the avalanche effect, where the modification of very few letters being encrypted causes a big change in output; or conversely, drastically different strings produce similar hash values. This effect causes hash values to not give any information regarding the input string, such as its original length. In addition, SHAs are also used to detect the tampering of data by attackers, where if a text file is slightly changed and barely noticeable, the modified file's hash value will be different than the original file's hash value, and the tampering will be rather noticeable.



A small tweak in the original data produces a drastically different encrypted output. This is called the avalanche effect [1] .

SHA Characteristics

Cryptographic hash functions are utilized in order to keep data secured by providing three fundamental safety characteristics: pre-image resistance, second pre-image resistance, and collision resistance.

The cornerstone of cryptographic security lies in the provision of **pre-image resistance**, which makes it hard and time-consuming for an attacker to find an original message, m,*m*, given the respective hash value, h_m*hm*. This security is provided by the nature of one-way functions, which is a key component of SHA. Pre-image resistance is necessary to ward off brute force attacks from powerful machines.

One-way Function

Alice and Bob are pen pals who share their thoughts via mail. When Alice visited Bob, she gave him a phone book of her city. In order to keep their messages safe from intruders, Alice tells Bob that she will encrypt the message. She tells Bob that he will find a bunch of numbers on every letter, and each sequence of numbers represents a phone number. Bob's job is to find the phone number in the book and write down the first letter of the person's last name. With this function, Bob is to decrypt the entire message.

To decrypt the message, Bob has to read the entire phone book to find all the numbers on the letter, whereas Alice can quickly find the letters and their respective phone numbers in order to encrypt her message. For this reason, before Bob is able to decrypt the message by hand, Alice can re-hash the message and keep the data secure. This makes Alice's algorithm a one-way function[2].

The second safety characteristic is called **second pre-image resistance**, granted by SHA when a message is known, m_1*m*1, yet it's hard to find another message, m_2*m*2, that hashes to the same value: H_{m_1} = H_{m_2}*Hm*1=*Hm*2. Without this characteristic, two different passwords would yield the same hash value, deeming the original password unnecessary in order to access secured data.

The last safety characteristic is **collision resistance**, which is provided by algorithms that make it extremely hard for an attacker to find two completely different messages that hash to the same hash value: H_{m_1} = H_{m_2}*Hm*1=*Hm*2. In order to provide this characteristic, there must be a similar number of possible inputs to possible outputs, as more inputs than outputs, by the pigeonhole principle, will definitively incur potential collisions. For this reason, collision resistance is necessary, as it implies that finding two inputs that hash to the same hash value is extremely difficult. Without collision resistance, digital signatures can be compromised as finding two messages that produce the same hash value may make users believe two documents were signed by two different people when one person was able to produce a different document with the same hash value.

Recent cryptographic functions have stronger security characteristics to block off recently developed techniques such as length extension attacks, where given a hash value, hash(m)*hash*(*m*), and the length of the original message, m*m*, an attacker can find a message, m'*m*', and calculate the hash value of the concatenation of the original message and the new message: hash\ (m||m')*hash* (*m*||*m*').

As a general guideline, a hash function should be as seemingly random as possible while still being deterministic and fast to compute.

SHA-1

Secure Hash Algorithm 1, or SHA-1, was developed in 1993 by the U.S. government's standards agency National Institute of Standards and Technology (NIST). It is widely used in security applications and protocols, including TLS, SSL, PGP, SSH, IPsec, and S/MIME.

SHA-1 works by feeding a message as a bit string of length less than 2^{64}264 bits, and producing a 160-bit hash value known as a *message digest*. Note that the message below is represented in hexadecimal notation for compactness.

There are two methods to encrypt messages using SHA-1. Although one of the methods saves the processing of sixty-four 32-bit words, it is more complex and time-consuming to execute, so the simple method is shown in the example below. At the end of the execution, the algorithm outputs blocks of 16 words, where each word is made up of 16 bits, for a total of 256 bits.

Pseudocode

Suppose the message 'abc' was to be encoded using SHA-1, with the message 'abc' in binary being

01100001 01100010 01100011

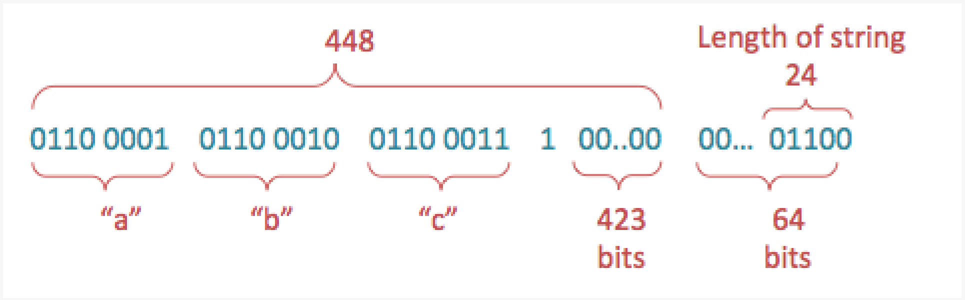
and that in hex being

616263.

1) The first step is to initialize five random strings of hex characters that will serve as part of the hash function (shown in hex):

H0 = 67DE2A01
H1 = BB03E28C
H2 = 011EF1DC
H3 = 9293E9E2
H4= CDEF23A9

2) The message is then padded by appending a 1, followed by enough 0s until the message is 448 bits. The length of the message represented by 64 bits is then added to the end, producing a message that is 512 bits long:



Padding of string "abc" in bits, finalized by the length of the string, which is 24 bits.

3) The padded input obtained above, MM, is then divided into 512-bit chunks, and each chunk is further divided into sixteen 32-bit words, W_0 … W_{15}$W0…W15$. In the case of 'abc', there's only one chunk, as the message is less than 512-bits total.

4) For each chunk, begin the 80 iterations, ii, necessary for hashing (80 is the determined number for SHA-1), and execute the following steps on each chunk, Mn:

- For iterations 16 through 79, where $16 \leq I \leq 79$, perform the following operation:

W(i) = $S^1(W(i-3) \oplus W(i-8) \oplus W(i-14) \oplus W(i-16))$, where XOR, or $\oplus$, is represented by the following comparison of inputs x and *y*:

x	y	Output
0	0	0
1	0	1
0	1	1
1	1	0

- For example, when *i* is 16, the words chosen are $W(13), W(8), W(2), W(0)$, and the output is a new word, $W(16)$, so performing the XOR, or $\oplus$, operation on those words will give this:

$W(0)$	01100001 01100010 01100011 10000000
$W(2)$	00000000 00000000 00000000 00000000
$W(8)$	00000000 00000000 00000000 00000000
$W(13)$	00000000 00000000 00000000 00000000
	$\oplus$
$W(16)$	01100001 01100010 01100011 10000000

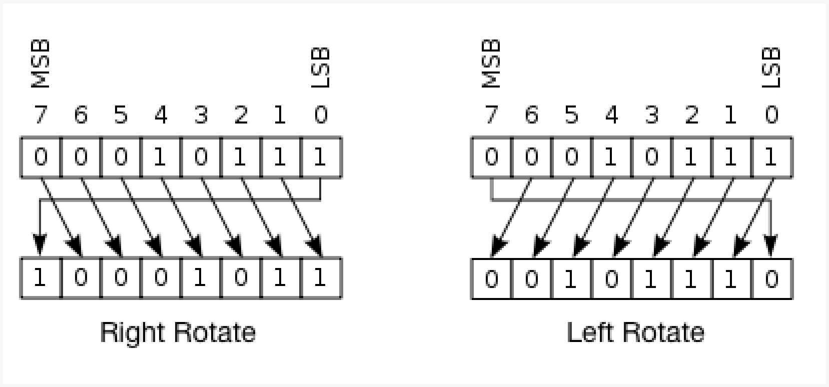
Circular Shift Operation

Now, the circular shift operation $Sn(X)$ on the word X by n bits, n being an integer between 0 and 32, is defined by

$$S^n(X) = (X << n) \textbf{ OR } (X >> 32-n),$$

where $X<<n$ is the **left-shift** operation, obtained by discarding the leftmost nn bits of X and padding the result with n zeroes on the right.

$X >> 32-n$ is the **right-shift** operation obtained by discarding the rightmost n bits of X and padding the result with nn zeroes on the left. Thus $S^n(X)$ is equivalent to a circular shift of X by n positions, and in this case the circular left-shift is used.



So, a left shift $S^n(W(i))$, where $W(i)$ is 10010, would produce 01001, as the rightmost bit 00 is shifted to the left side of the string. Therefore, $W(16)$ would end up being

11000010 11000100 11000111 000000000.

5) Now, store the hash values defined in step 1 in the following variables:

$$A = H_0$$
$$B = H_1$$
$$C = H_2$$
$$D = H_3$$
$$E = H_4.$$

6) For 8080 iterations, where $0 \leq I \leq 79$, compute

$$TEMP = S^5 * (A) + f(i;B,C,D) + E + W(i) + K(i).$$

See below for details on the logical function, f, and on the values of K(i). Reassign the following variables:

$$E = D$$
$$D = C$$
$$C = S^{30}(B)$$

$$B = A$$
$$A = TEMP.$$

7) Store the result of the chunk's hash to the overall hash value of all chunks, as shown below, and proceed to execute the next chunk:

$$H_0 = H_0+A$$
$$H_1 = H_1+B$$
$$H_2 = H_2+C$$
$$H_3 = H_3+D$$
$$H_4 = H_4+E.$$

8) As a final step, when all the chunks have been processed, the message digest is represented as the 160-bit string comprised of the **OR** logical operator, $\vee$, of the 5 hashed values:

$$HH = S^{128}(H_0)\vee S^{96}(H_1)\vee S^{64}(H_2)\vee S^{32}(H_3)\vee H4.$$

So, the string 'abc' becomes represented by a hash value akin to **a9993e364706816aba3e25717850c26c9cd0d89d**.

If the string changed to 'abcd', for instance, the hashed value would be drastically different so attackers cannot tell that it is similar to the original message. The hash value for 'abcd' is **81fe8bfe87576c3ecb22426f8e57847382917acf**.

Functions used in the algorithm

A sequence of logical functions are used in SHA-1, depending on the value of i, where $0 \leq i \leq 79$, and on three 32-bit words B, C, and D, in order to produce a 32-bit output. The following equations describe the logical functions, where $\neg$ is the logical *NOT*, $\vee$ is the logical *OR*, $\wedge$ is the logical *AND*, and $\oplus$ is the logical XOR:

$$f(i;B,C,D)=(B\wedge C)\vee((\neg B)\wedge D) \qquad \text{for} 0\geq i \geq 19$$
$$f(i;B,C,D)=B\oplus C\oplus D \qquad \text{for} 20\geq i \geq 39$$
$$f(i;B,C,D)=(B\wedge C)\vee(B\wedge D)\vee(C\wedge D) \qquad \text{for} 40\geq i \geq 59$$
$$f(i;B,C,D)=B \oplus C\oplus D \qquad \text{for} 60\geq i \geq 79.$$

Additionally, a sequence of constant words, shown in hex below, is used in the formulas:

$$K(i) = 5A827999, \qquad \text{where} 0\leq i\leq 19$$
$$K(i) = 6ED9EBA1, \qquad \text{where} 20\leq i\leq 39$$
$$K(i) = 8F1BBCDC, \qquad \text{where} 40\leq i\leq 59$$
$$K(i) = CA62C1D6, \qquad \text{where} 60\leq i\leq 79.$$

Albeit SHA-1 is still widely used, cryptanalysts in 2005 were able to find vulnerabilities on this algorithm that detrimentally compromised its security. These vulnerabilities came in the form of an algorithm that speedily finds collisions with different inputs, meaning that two distinct inputs map to the same digest.

As of 2010, many organizations have recommended its replacement by SHA-2 or SHA-3. Companies like Microsoft, Google, or Mozilla have announced that their browsers will stop accepting SHA-1 encryption certificates by 2017.

SHA-2

Due to the exposed vulnerabilities of SHA-1, cryptographers modified the algorithm to produce SHA-2, which consists of not one but two hash functions known as SHA-256 and SHA-512, using 32- and 64-bit words, respectively. There are additional truncated versions of these hash functions, known as SHA-224, SHA-384, SHA-512/224, and SHA-512/256, which can be used for either part of the algorithm.

SHA-1 and SHA-2 differ in several ways; mainly, SHA-2 produces 224- or 256-sized digests, whereas SHA-1 produces a 160-bit digest; SHA-2 can also have block sizes that contain 1024 bits, or 512 bits, like SHA-1.

Brute force attacks on SHA-2 are not as effective as they are against SHA-1. A brute force search for finding a message that corresponds to a given digest of length L using brute force would require 2^L evaluations, which makes SHA-2 a lot safer against these kinds of attacks.

Common Attacks

Cryptography wouldn't be as quickly developed if it weren't for the attacks that compromise their effectiveness. One of the most common attacks is known as the prime age attack, where pre-computed tables of solutions are used in a brute-force manner in order to crack passwords. The solution against these kinds of attacks is to compose a hash function that would take an attacker an exorbitant amount of resources, such as millions of dollars or decades of work, to find a message corresponding to a given hash value.

Most attacks penetrating SHA-1 are collision attacks, where a non-sensical message produces the same hash value as the original message. Generally, this takes time proportional to $2^{n/2}$ to complete, where n is the length of the message. This is the reason the message digests have increased in length from 160-bit digests in SHA-1 to 224- or 256-bit digests in SHA-2.

Other attacks exist that attempt to exploit mathematical properties in order to crack hash functions. Amongst these is the birthday attack, where higher likelihood of collisions are found when using random attacks with a fixed number of letter combinations (see the pigeonhole principle), or the rainbow table attack, where a pre-computed hash table is used to reverse a hash function in order to crack passwords.

PRINCIPLES OF PUBLIC KEY CRYPTOGRAPHY

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

- Key distribution under symmetric key encryption requires either (1) that two communicants already share a key, which someone has been distributed to them or (2) the use of a key distribution center.
- Digital signatures.

1. Public key cryptosystems

Public key algorithms rely on one key for encryption and a different but related key for decryption.

These algorithms have the following important characteristics:

- It is computationally infeasible to determine the decryption key given only the knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristic:

- Either of the two related keys can be used for encryption, with the other used for decryption.

The essential steps are the following:

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

- Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private.

- If A wishes to send a confidential message to B, A encrypts the message using B"s public key.

- When B receives the message, it decrypts using its private key. No other recipient can decrypt the message because only B knows B"s private key.

With this approach, all participants have access to public keys and private keys are generated locally by each participant and therefore, need not be distributed. As long as a system controls its private key, its incoming communication is secure.

Let the plaintext be X=[X1, X2, X3, …,Xm] where m is the number of letters in some finite alphabets. Suppose A wishes to send a message to B. B generates a pair of keys: a public key KU_b and a private key KR_b. KR_b is known only to B, whereas KU_b is publicly available and therefore accessible by A.

With the message X and encryption key KU_b as input, A forms the cipher text

Y=[Y1, Y2, Y3, … Yn]., i.e., Y=E KU_b(X)

The receiver can decrypt it using the private key KR_b. i.e., X=D KR_b(). The encrypted message serves as a **digital signature.**

It is important to emphasize that the encryption process just described does not provide confidentiality. There is no protection of confidentiality because any observer can decrypt the message by using the sender‟s public key.

It is however, possible to provide both the authentication and confidentiality by a double use of the public scheme.

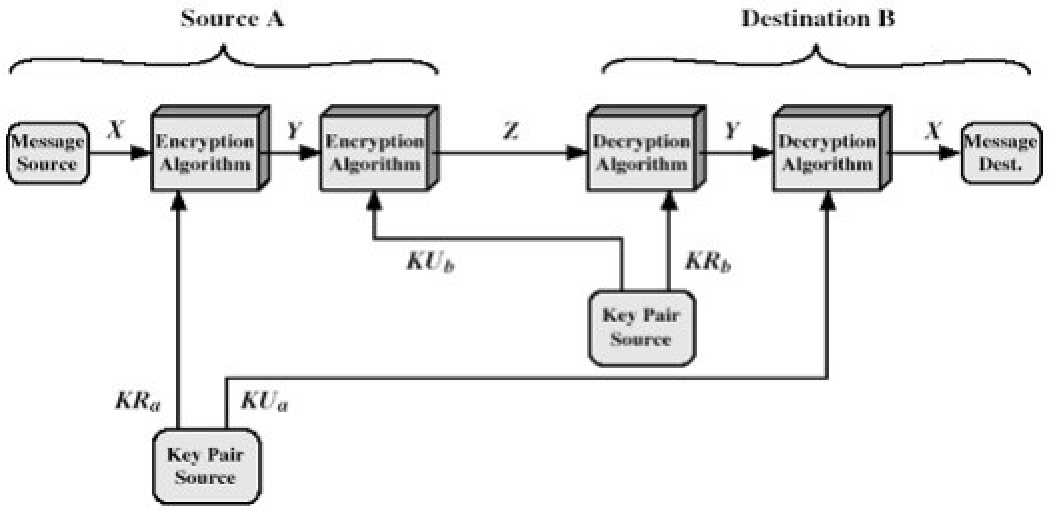


Fig.2.7.1.1 Public Key Cryptosystem

Ciphertext Z = EKU_b [EKR_a (X)]

Plaintext X = EKU_a[EKR_b (Y)]

Initially, the message is encrypted using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus confidentiality is provided.

2 Requirements for public key cryptography

It is computationally easy for a party B to generate a pair [KU_b , KR_b].

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

It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: $M = DKR_b (C) = DKR_b [EKU_b (M)]$

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

It is computationally infeasible for an opponent, knowing the public key KU_b, and a ciphertext C, to recover the original message M.

The encryption and decryption functions can be applied in either order: $M = EKU_b [DKR_b (M) = DKU_b [EKR_b (M)]$

Public key cryptanalysis

Public key encryption scheme is vulnerable to a brute force attack. The counter measure is to use large keys.

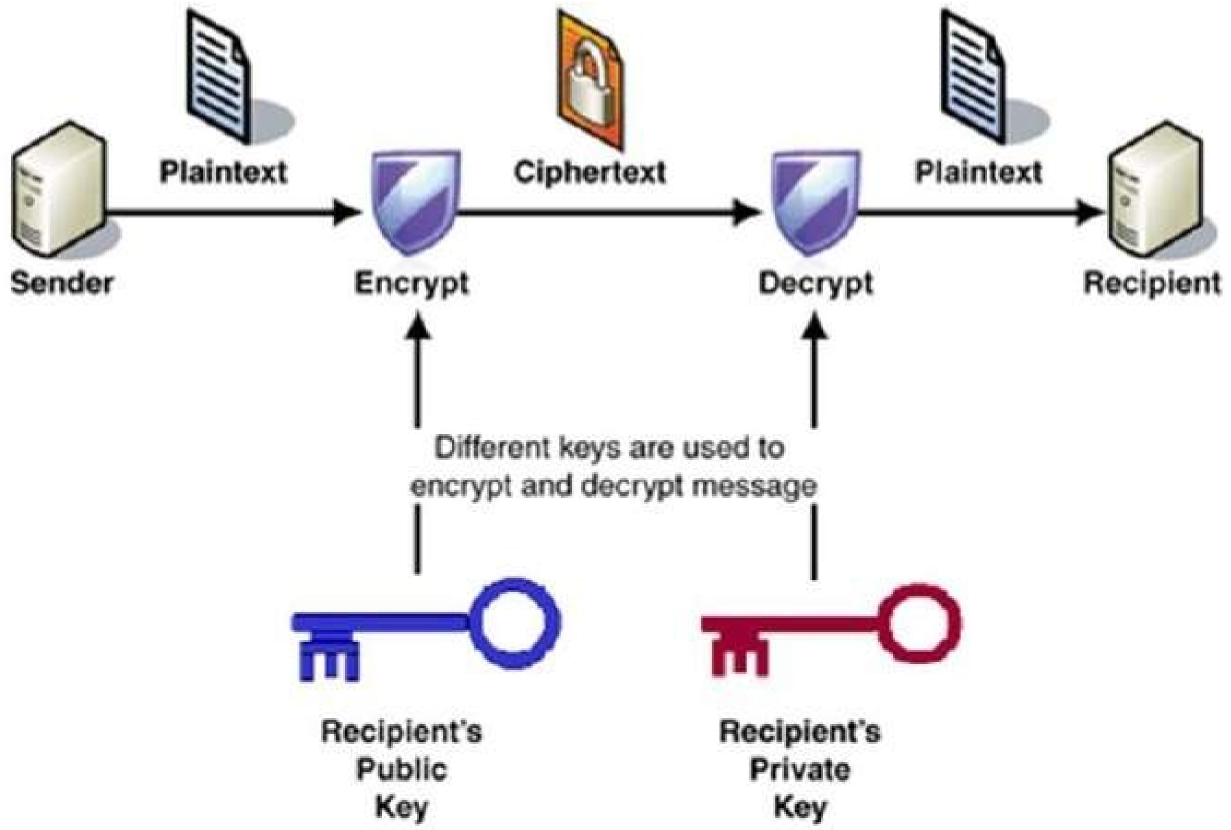
Public Key Cryptography

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.

There are three types of Public Key Encryption schemes. We discuss them in following sections −

RSA Cryptosystem

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 coprime.
- **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 −

$ed = 1 \bmod (p - 1)(q - 1)$

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 −

$de = 29 \times 5 = 145 = 1 \bmod 72$

- 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 −

$C = P^e \bmod n$

- 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 −

$C = 10^5 \bmod 91$

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.

Plaintext = C^d mod n

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

Plaintext = 82^{29} mod 91 = 10

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.

If either of these two functions are proved non one-way, then RSA will be broken. In fact, if a technique for factoring efficiently is developed then RSA will no longer be safe.

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.

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 mod n.

    For example, 3 is generator of group 5 (Z_5 = {1, 2, 3, 4}).

N	**3^n**	**3^n mod 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 −

$y = g^x \bmod p$

- **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 71, so we choose x = 5. The value y is then computed as follows −

$y = 6^5 \bmod 17 = 7$

- Thus the private key is 62 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 −

$C1 = g^k \bmod p$
$C2 = (P^* y^k) \bmod p$

- 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 −

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

- 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 −

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

- 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 × 9) mod 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 mod 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.

Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is a term used to describe a suite of cryptographic tools and protocols whose security is based on special versions of the discrete logarithm problem. It does not use numbers modulo p.

ECC is based on sets of numbers that are associated with mathematical objects called elliptic curves. There are rules for adding and computing multiples of these numbers, just as there are for numbers modulo p.

ECC includes a variants of many cryptographic schemes that were initially designed for modular numbers such as ElGamal encryption and Digital Signature Algorithm.

It is believed that the discrete logarithm problem is much harder when applied to points on an elliptic curve. This prompts switching from numbers modulo p to points on an elliptic curve. Also an equivalent security level can be obtained with shorter keys if we use elliptic curve-based variants.

The shorter keys result in two benefits −

- Ease of key management
- Efficient computation

These benefits make elliptic-curve-based variants of encryption scheme highly attractive for application where computing resources are constrained.

RSA and ElGamal Schemes – A Comparison

Let us briefly compare the RSA and ElGamal schemes on the various aspects.

RSA	**ElGamal**
It is more efficient for encryption.	It is more efficient for decryption.
It is less efficient for decryption.	It is more efficient for decryption.
For a particular security level, lengthy keys are required in RSA.	For the same level of security, very short keys are required.
It is widely accepted and used.	It is new and not very popular in market.

Digital Signatures

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

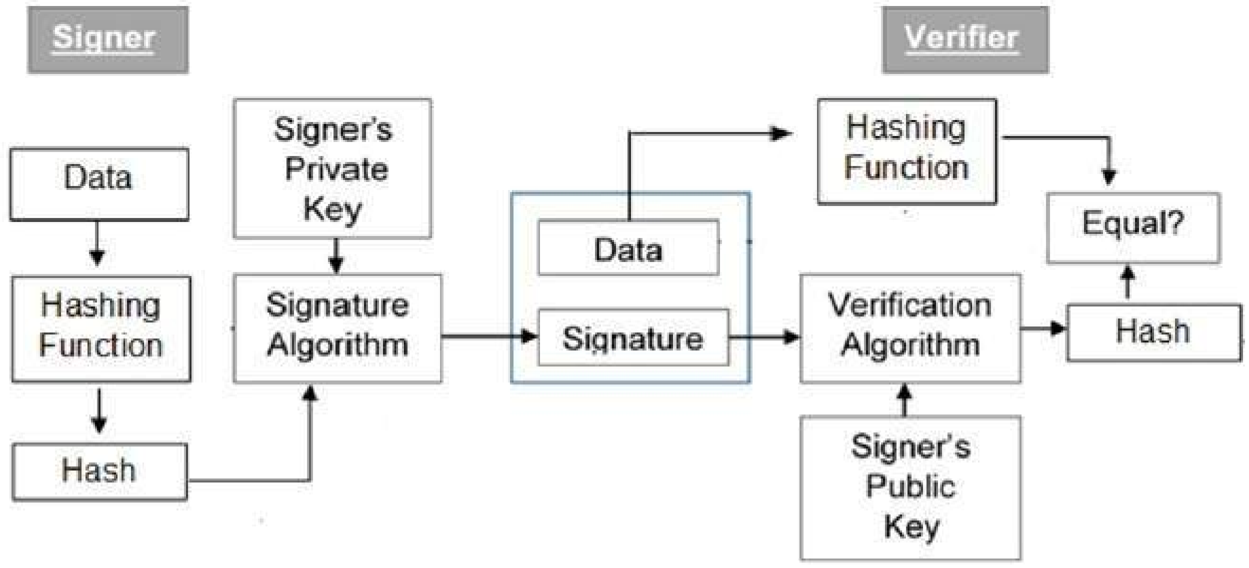
Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature

As mentioned earlier, 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.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely − Privacy, Authentication, Integrity, and Non-repudiation.

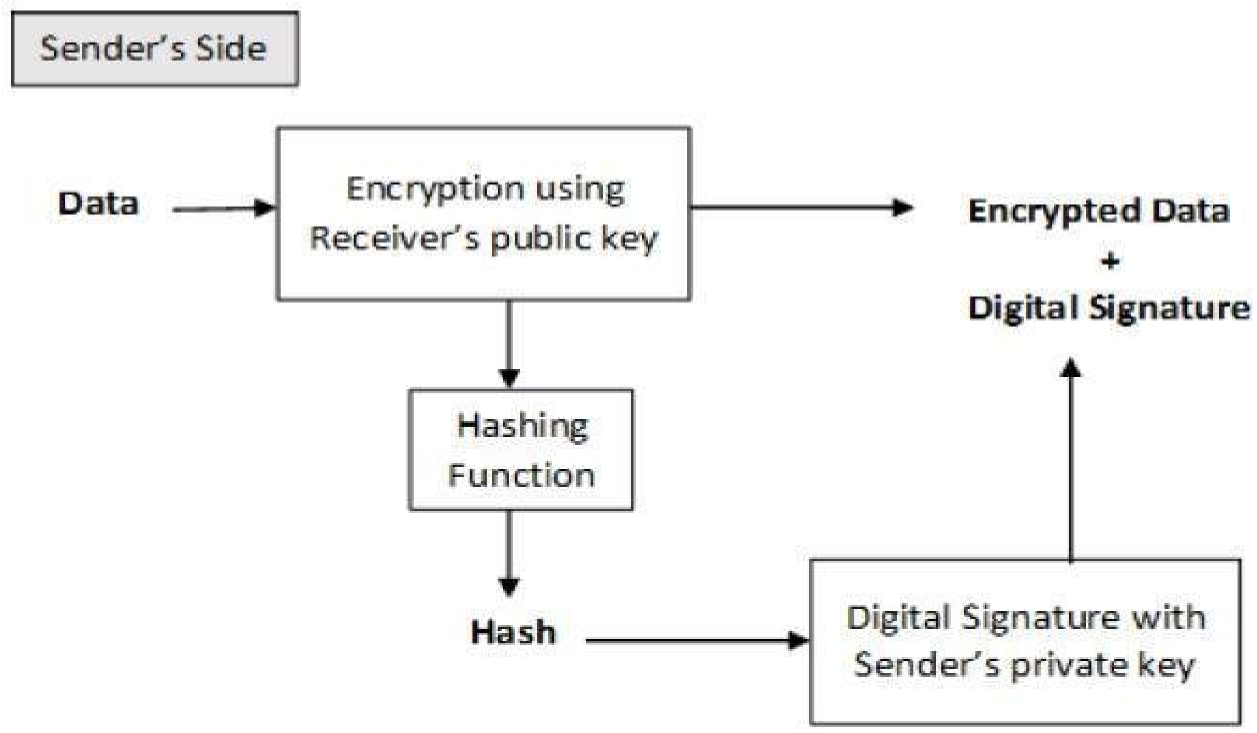
Encryption with Digital Signature

In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.

This can archive by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are **two possibilities, sign-then-encrypt** and **encrypt-then-sign**.

However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration −


Sender's Side
Data
Encryption using Receiver's public key
Encrypted Data + Digital Signature
Hashing Function
Hash
Digital Signature with Sender's private key

The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender's public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

Key Management and Distribution

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

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

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

e.g.

$$\text{Alice} \rightarrow \text{Bob: } \{ \text{“Hello World”} \}\ k$$

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 sessions

  changes independently of session initiation and termination

Key Exchange

e.g.

Alice and Bob want to communicate securely

The goal is for Alice and Bob to communicate secretly.

1. Key cannot be transmitted in the clear
2. Bob and Alice may decide to trust a third party
3. The cryptosystems and protocols are known. Only the keys are secret.

Classical Cryptographic Key Exchange & Authentication

Trusted third party

1. Alice → Cathy: { request for session key to Bob} K_{Alice}
2. Cathy → Alice: { $K_{Session}$ }K_{Alice} || { KSession } K_{Bob}
3. Alice → Bob: { $K_{Session}$ } K_{Bob}

Alice wants to talk to Bob

Alice & Cathy share a secret key

Bob & Cathy share a secret key

Goal: Alice and Bob share a secret key

Public Key Cryptographic Key Exchange and Authentication

Again, Alice wants to secretly communicate with Bob

Alice → Bob: { $K_{Session}$ } $K_{Bob_{Pub}}$

Bob decodes and away they go BUT ……

How does Bob know that the $K_{Session}$ came from Alice?

Alice → Bob: { { $K_{Session}$ } $K_{Alice_{\mathrm{Pr}\,i}}$ } $K_{Bob_{Pub}}$

Suppose that Eve is listing to Alice

Alice → Peter: { “send me Bob’s public key” } - Eve hears

Eve → Peter: { “send me Bob’s public key” }

Peter → Eve: { $Key_{Bob_{Pub}}$ }

Eve → Alice: { $Key_{Eve_{Pub}}$ }

Alice → Bob: { $K_{Session}$ } $Key_{Eve_{Pub}}$ - This is intercepted by Eve

Eve → Bob: { $K_{Session}$ } $K_{Bob_{Pub}}$

This is an example of a man-in-the-middle attack

Key distribution

Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data. Some sort of mechanism or protocol is needed to provide for the secure distribution of keys.

- o **Symmetric Key Distribution Using Symmetric Encryption**
  - A Key Distribution Scenario ierarchical Key Control Session Key Lifetime
  - A Transparent Key Control Scheme Decentralized Key Control Controlling Key Usage
- o **Symmetric Key Distribution Using Asymmetric Encryption**
  - Simple Secret Key Distribution
  - Secret Key Distribution with Confidentiality and Authentication A Hybrid Scheme
- o **Distribution Of Public Keys**
  - Public Announcement of Public Keys Publicly Available Directory
  - Public-Key Authority Public-Key Certificates
- o **X.509 Certificates**
  - Certificates
  - X.509 Version 3
- o **Public-Key Infrastructure**

- PKIX Management Functions PKIX Management Protocols

KEY POINTS

◆ Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data. Some sort of mechanism or protocol is needed to provide for the secure distribution of keys.

◆ Key distribution often involves the use of master keys, which are infrequently used and are long lasting, and session keys, which are generated and distributed for temporary use between two parties.

◆ Public-key encryption schemes are secure only if the authenticity of the public key is assured. A public-key certificate scheme provides the necessary security.

◆ X.509 defines the format for public-key certificates. This format is widely used in a variety of applications.

◆ A public-key infrastructure (PKI) is defined as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

◆ Typically, PKI implementations make use of X.509 certificates.

UNIT - III

KERBEROS

Kerberos is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workst ation cannot be trusted to identify its users correctly to network services.In particular, the followi ng three threats exist:

1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.

2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.

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

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elabo- rate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes described in this book, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

Two versions of Kerberos are in common use. Version 4 [MILL88, STEI88] implementations still exist. Version 5 [KOHL94] corrects some of the security deficiencies of version 4 and has been issued as a proposed Internet Standard (RFC 4120).

If a set of users is provided with dedicated personal computers that have no network connections, then a user's resources and files can be protected by physically securing each personal computer. When these users instead are served by a centralized time sharing system, the time-sharing operating system must provide the security. The operating system can enforce access control policies based on user identity and usethe logon procedure to identify users.

Today, neither of these scenarios is typical. More common is a distributed architecture consisting of
dedicated user workstations (clients) and distributed or centralized servers. In this environment, three approaches to security can be envisioned.

1. Rely on each individual client workstation to assure the identity of its user or
users and rely on each server to enforce a security policy based on user identi- fication (ID).

2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.

3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

In a small, closed environment in which all systems are owned and operated bya single organization, the first or perhaps the second strategy may suffice. But in a more open environment in which network connections to other machines are supported, the third approach is needed to protect user information and resources housed at the server. Kerberos supports this third approach. Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

The first published report on Kerberos [STEI88] listed the following requirements.

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.
- **Transparent:** Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol based on that proposed by Needham and Schroeder [NEED78].

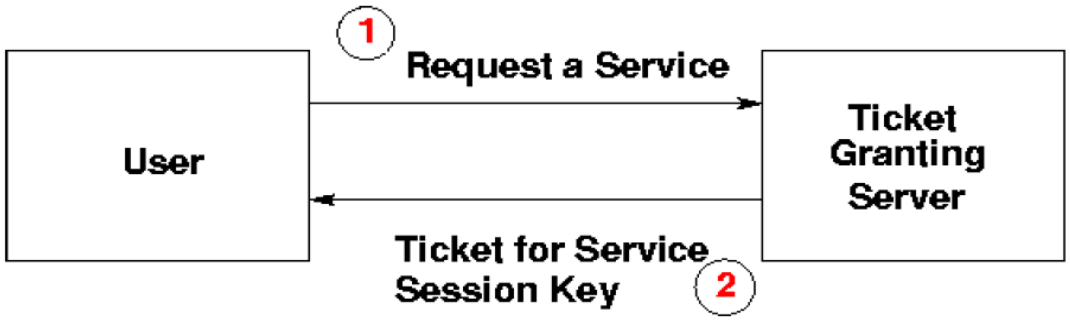
It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure.

Example:

Ticket

is an unforgeable, non-replayable, authenticated object

names the user & service user allowed to use



Flow of Information

1. user identifies self to Kerberos Server.

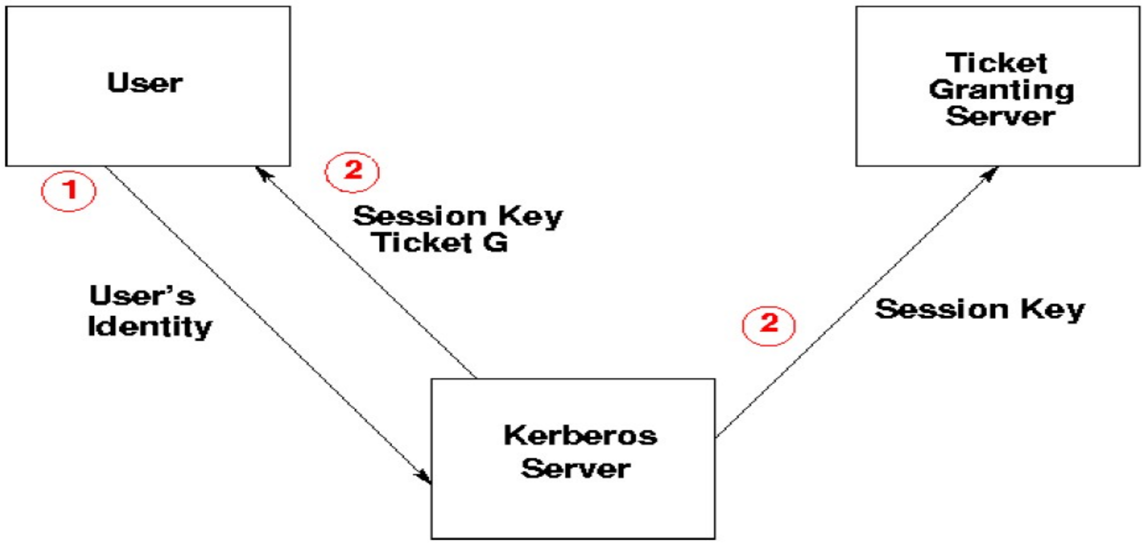
2. Kerberos Server verifies the user is authorized.

3. Kerberos Server sends a Session Key SG for use in communications with the TG server and a Ticket TG for the TG server encrypted with the user's password.

4. Kerberos Server sends a copy of the Session Key SG to the TG server, the identity of the user encrypted with a Key shared by the Kerberos server and the TG server.

Note:

User's password stored at Kerberos Server and not passed over the networks.



Ticket contains

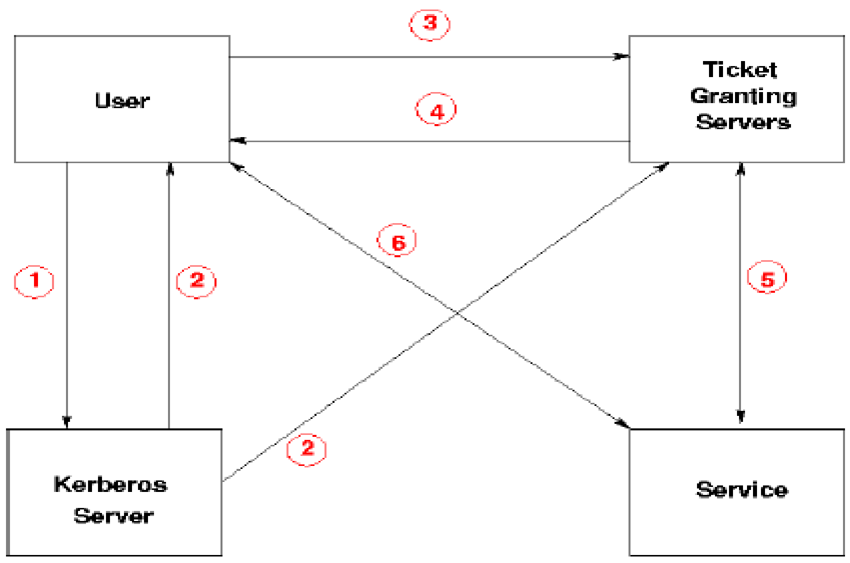
User's authenticated identity

identification of requested service

rights w.r.t. services

Session Key

expiration date of ticket


3
User
Ticket Granting Servers
4
6
1
2
5
2
Kerberos Server
Service

1. Authentication
2. Ticket Authorization – Authorization Key
3. Server Access Request
4. Service Ticket
5. Unique Keys between TG Server and Service
6. Service Request

Characteristics

1. No passwords communicated on the network
    initial password passed by snail mail
2. Cryptographic protection against spoofing
3. Limited period of validity
4. Timestamps to prevent replay attacks
5. Mutual authentication

Issues

1. continuous availability of a trusted TG server.
2. Authenticity of servers requires a trusted relationship between that TG server and every server.
3. Requires timely transactions
4. Subverted workstation can save and later replay user passwords.
5. Password guessing works
    intercept ticket
6. Does not scale well.
7. Complete solution
    all applications must use Kerberos
    authentication

X.509 CERTIFICATES

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

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates of the type discussed in Section 14.3. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

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

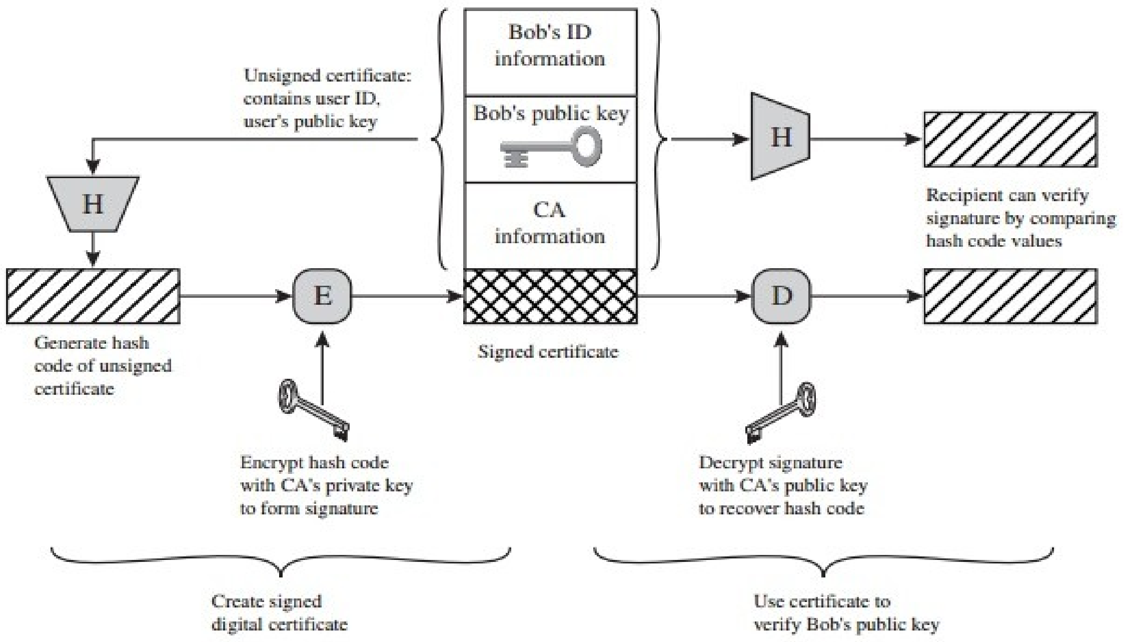
X.509 was initially issued in 1988. The standard was subsequently revised to address some of the security concerns documented in [IANS90] and [MITC90]; a revised recommendation was issued in 1993. A third version was issued in 1995 and revised in 2000.

X.509 is based on the use of public key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA.The digital signature scheme is assum ed
to require the use of a hash function. Again, the standard does not dictate a specific hash algorithm. The 1988 recommendation included the description of a recommended hash algorithm; this algorithm has since been shown to be insecure and was dropped from the 1993 recommendation. Figure 14.13 illustrates the generation of a public-key certificate.

Certificates

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

Figure 14.14a shows the general format of a certificate, which includes the fol- lowing elements.



Unsigned certificate: contains user ID, user's public key
Bob's ID information
Bob's public key
CA information
H
H
Recipient can verify signature by comparing hash code values
E
D
Generate hash code of unsigned certificate
Signed certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 14.13 Public-Key Certificate Use

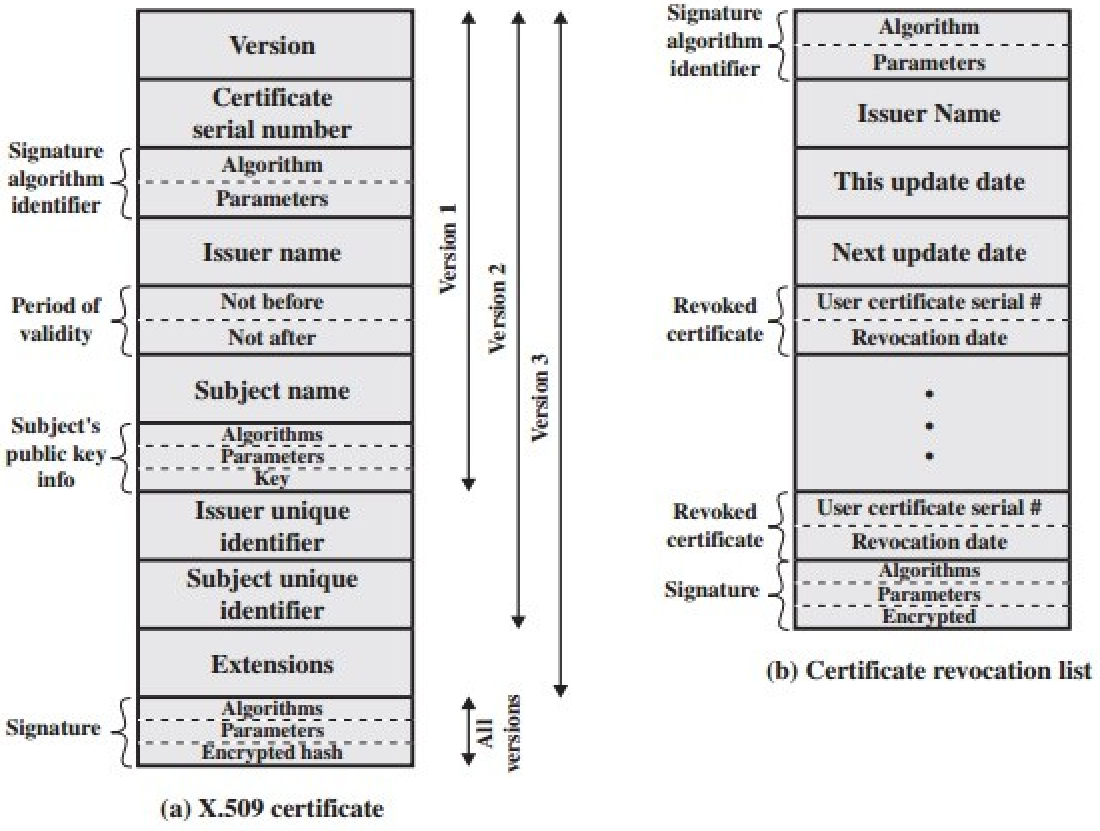


Figure 14.14 X.509 Formats

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the *issuer unique identifier* or *subject unique identifier* are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
- **Serial number:** An integer value unique within the issuing CA that is unam biguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate togethe r with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 is the name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier:** An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

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

The standard uses the following notation to define a certificate:

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

where

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

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

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificate

CA = name of certificate authority

UCA = optional unique identifier of the CA

A = name of user A

UA = optional unique identifier of the user A

Ap = public key of user A

TA = period of validity of the certificate

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

OBTAINING A USER'S CERTIFICATE User certificates generated by a CA have the following characteristics:

• Any user with access to the public key of the CA can verify the user public key that was certified.

• No party other than the certification authority can modify the certificate with- out this being detected.

Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

If all users subscribe to the same CA, then there is a common trust of that CA.
All user certificates can be placed in the directory for access by all users. In addition, a user can
transmit his or her certificate directly to other users. In either case, once B is in possession of A's
certificate, B has confidence that messages it encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable.

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each par- ticipating user must have a copy of the CA's own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

Now suppose that A has obtained a certificate from certification authority X1 and B has obtained a certificate from CA X2. If A does not securely know the public key of X2, then B's certificate, issued by X2, is useless to A. A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.

Step 1 A obtains from the directory the certificate of X2 signed by X1. Because A
securely knows X1's public key, A can obtain X2's public key from its certificate and verify it b
y means of X1's signature on the certificate.

Step 2 A then goes back to the directory and obtains the certificate of B signed by X2. Because A now has a trusted copy of X2's public key, A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

X1 << X2 >> X2 << B >>

In the same fashion, B can obtain A's public key with the reverse chain:

X2 << X1 >> X1 << A >>

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with *N* elements would be expressed as

X1 << X2 >> X2 << X3 >> Á X*N* << B >>

In this case, each pair of CAs in the chain (X*i*, X*i*+1) must have created certificates for each other.

All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public-key certificate. X.509 suggests that CAs be arranged in a hierarchy so that navigation is straight forward.

Figure 14.15, taken from X.509, is an example of such a hierarchy. The con- nected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates maintained in the directory for each CA entry. The directory entry for each CA includes two types of certificates:

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

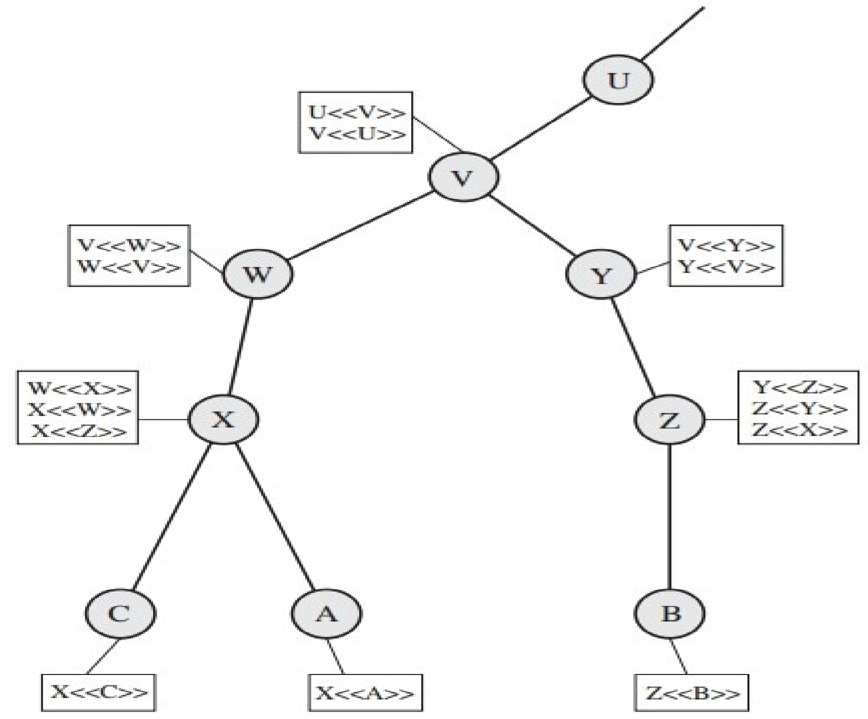


Figure 14.15 X.509 Hierarchy: A Hypothetical Example

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

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

When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key. Using this public key, A can send encrypted messages to B. If A wishes to receive encrypted messages back from B, or to sign messages sent to B, then B will require A's public key, which can be obtained from the following certification path:

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

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

REVOCATION OF CERTIFICATES

Recall from Figure 14.14 that each certificate includes a period of validity, much like a credit card. Typically, a new certifica te is issued just before the expiration of the old one. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.

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

2. The user is no longer certified by this CA. Reasons for this include that the subject's name h as changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.

3. The CA's certificate is assumed to be compromised.

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

Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes (Figure 14.14b) the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate. Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.

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

X.509 Version 3

The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. [FORD95] lists the following requirements not satisfied by version 2.

1. The subject field is inadequate to convey the identity of a key owner to a public-key user. X.509 names may be relatively short and lacking in obvious identification details that may be needed by the user.

2. The subject field is also inadequate for many applications, which typically recogni ze entities by an Internet e-mail address, a URL, or some other Internet-related identification.

3. There is a need to indicate security policy information. This enables a security application or function, such as IPSec, to relate an X.509 certificate to a given policy.

4. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.

5. It is important to be able to identify different keys used by the same owner at different times. This feature supports key lifecycle management: in particular, the ability to update key pairs for users and CAs on a regular basis or under exceptional circumstances.

Rather than continue to add fields to a fixed format, standards developers felt that a more flexible approach was needed. Thus, version 3 includes a number of optional extensions that may be added to the version 2 format. Each extension consists of an ext ension identifier, a criticality indicator, and an extension value. The criticality indicator indicates whether an extension can be safely ignored. If the indicator has a value of TRUE and an implementation does not recognize the extension, it must treat the certific ate as invalid.

The certificate extensions fall into three main categories: key and policy information, subject and issuer attributes, and certification path constraints.

KEY AND POLICY INFORMATION These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy. A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes:

- **Authority key identifier:** Identifies the public key to be used to verify the signature on this certificate or CRL. Enables distinct keys of the same CA to be differentiated. One use of this field is to handle CA key pair updating.

- **Subject key identifier:** Identifies the public key being certified. Useful for subj ect key pair updating. Also, a subject may have multiple key pairs and, correspondingly, differen t certificates for different purposes (e.g., digital signature and encryption key agreement).

- **Key usage:** Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used. May indicate one or more of the following: digital signature, nonrepudiation, key encryption, data encryption, key agreement, CA signature verification on certificates, CA signature verification on CRLs.

- **Private-key usage period:** Indicates the period of use of the private key corre- sponding to the public key. Typically, the private key is used over a different period from the validity of the public key. For example, with digital signature keys, the usage per iod for the signing private key is typically shorter than that for the verifying public key.

- **Certificate policies:** Certificates may be used in environments where multiple policies apply. This extension lists policies that the certificate is recognized as supporting, together with optional qualifier information.

- **Policy mappings:** Used only in certificates for CAs issued by other CAs. Policy mappings allow an issuing CA to indicate that one or more of that issuer's policies can be considered equivalent to another policy used in the subject CA's domain.

CERTIFICATE SUBJECT AND ISSUER ATTRIBUTES These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject to increase a certificate user's confidence that the certificate subject is a particular person or entity. For example, information such as postal address, position within a corporation, or picture image may be required.

The extension fields in this area include:

- **Subject alternative name:** Contains one or more alternative names, using any of a variety of forms. This field is important for supporting certain applications, such as electronic mail, EDI, and IPSec, which may employ their own name forms.
- **Issuer alternative name:** Contains one or more alternative names, using any of a variety of forms.
- **Subject directory attributes:** Conveys any desired X.500 directory attribute values for the subject of this certificate.

CERTIFICATION PATH CONSTRAINTS These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs. The constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.

The extension fields in this area include:

- **Basic constraints:** Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.
- **Name constraints:** Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.
- **Policy constraints:** Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certifi- cation path.

PUBLIC-KEY INFRASTRUCTURE

RFC 2822 (*Internet Security Glossary*) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the dri- ving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. This section describes the PKIX model.

Figure 14.16 shows the interrelationship among the key elements of the PKIX model. These elements are

- **End entity:** A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a pub- lic key certificate. End entities typically consume and/or support PKI-related services.
- **Certification authority (CA):** The issuer of certificates and (usually) certificat e revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.

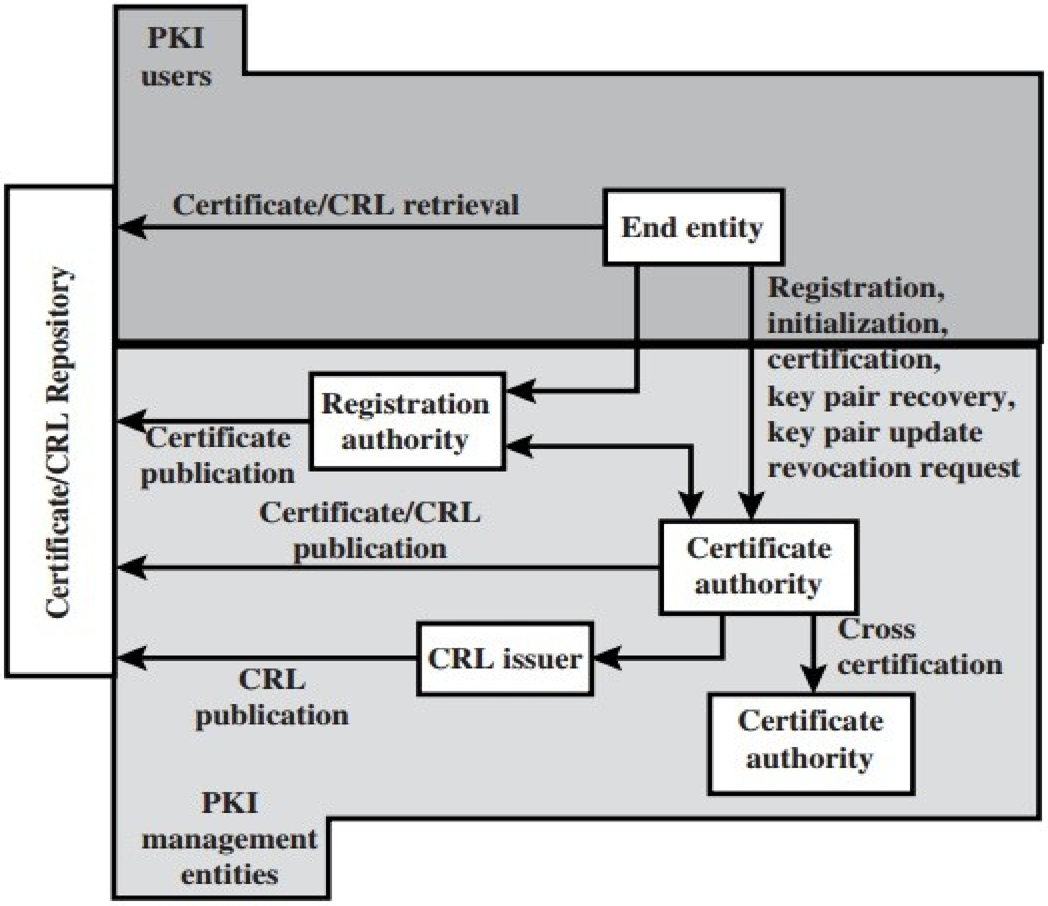

PKI users
Certificate/CRL retrieval
End entity
Registration, initialization, certification, key pair recovery, key pair update revocation request
Certificate/CRL Repository
Registration authority
Certificate publication
Certificate/CRL publication
Certificate authority
CRL issuer
Cross certification
CRL publication
Certificate authority
PKI management entities

Figure 14.16 PKIX Architectural Model

- **Registration authority (RA):** An optional component that can assume a numbe r of administrative functions from the CA. The RA is often associated with the end entity registrati on process but can assist in a number of other areas as well.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

PKIX Management Functions

PKIX identifies a number of management functions that potentially need to be sup- ported by management protocols. These are indicated in Figure 14.16 and include the following:

- **Registration:** This is the process whereby a user first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate or cer- tificates for that user. Registration begins the process of enrolling in a PKI. Registration usually involves some offline or online procedure for mutual authentication. Typically, the end entity is issued one or more shared secret keys used for subsequent authentication.

Initialization: Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure. For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths.

- **Certification:** This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.

- **Key pair recovery:** Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the nec- essary decryption keys when normal access to the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data. Loss of access to the decryption key can result from forgotten passwords/PINs, corrupted disk drives, damage to hardware tokens, and so on. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the end entity's certificate).

- **Key pair update:** All key pairs need to be updated regularly (i.e., replaced with a new key pair) and new certificates issued. Update is required when the certificate lifetime expir eand as a result of certificate revocation.

- **Revocation request:** An authorized person advises a CA of an abnormal situati on requiring certificate revocation. Reasons for revocation include private-key compromise, change in affiliation, and name change.

- **Cross certification:** Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

PKIX Management Protocols

The PKIX working group has defines two alternative management protocols between PKIX entities that support the management functions listed in th e preced- ing subsection. RFC 2510 defines the certificate management protocols (CMP). Within CMP, each of the management functions is explicitly identified by spec ific protocol exchanges. CMP is designed to be a flexible protocol able to accommodate a variety of technical, operational, and business models.

RFC 2797 defines certificate management messages over CMS (CMC), where CMS refers to RFC 2630, cryptographic message syntax. CMC is built on earlier work and is intended to leverage existing implementations. Although all of the PKIX functions are supported, the functions do not a ll map into specific protocol exchanges.

ELECTRONIC MAIL SECURITY

1 PRETTY GOOD PRIVACY (PGP)

PGP provides the confidentiality and authentication service that can be used for electronic mail and file storage applications. The steps involved in PGP are:

Select the best available cryptographic algorithms as building blocks.

Integrate these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.

Make the package and its documentation, including the source code, freely available via the internet, bulletin boards and commercial networks.

Enter into an agreement with a company to provide a fully compatible, low cost commercial version of PGP.

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

It is available free worldwide in versions that run on a variety of platform.

It is based on algorithms that have survived extensive public review and are considered extremely secure.

e.g., RSA, DSS and Diffie Hellman for public key encryption CAST-128, IDEA and 3DES for conventional encryption SHA-1 for hash coding.

It has a wide range of applicability.

It was not developed by, nor it is controlled by, any governmental or standards organization.

Operational description

The actual operation of PGP consists of five services: authentication, confidentiality, compression, e-mail compatibility and segmentation.

1. Authentication

The sequence for authentication is as follows:

§ The sender creates the message

§ SHA-1 is used to generate a 160-bit hash code of the message

§ The hash code is encrypted with RSA using the sender‟s private key and the result is prepended to the message

§ The receiver uses RSA with the sender‟s public key to decrypt and recover the hash code.

The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

2. Confidentiality

Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the conventional encryption algorithm CAST-128 may be used. The 64-bit cipher feedback (CFB) mode is used.

In PGP, each conventional key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus although this is referred to as **a session key**, it is in reality a **one time key**. To protect the key, it is encrypted with the receiver‟s public key.

The sequence for confidentiality is as follows:

The sender generates a message and a random 128-bit number to be used as a session key for this message only.

The message is encrypted using CAST-128 with the session key.

The session key is encrypted with RSA, using the receiver‟s public key and is prepended to the message.

The receiver uses RSA with its private key to decrypt and recover the session key.

The session key is used to decrypt the message.

Confidentiality and authentication

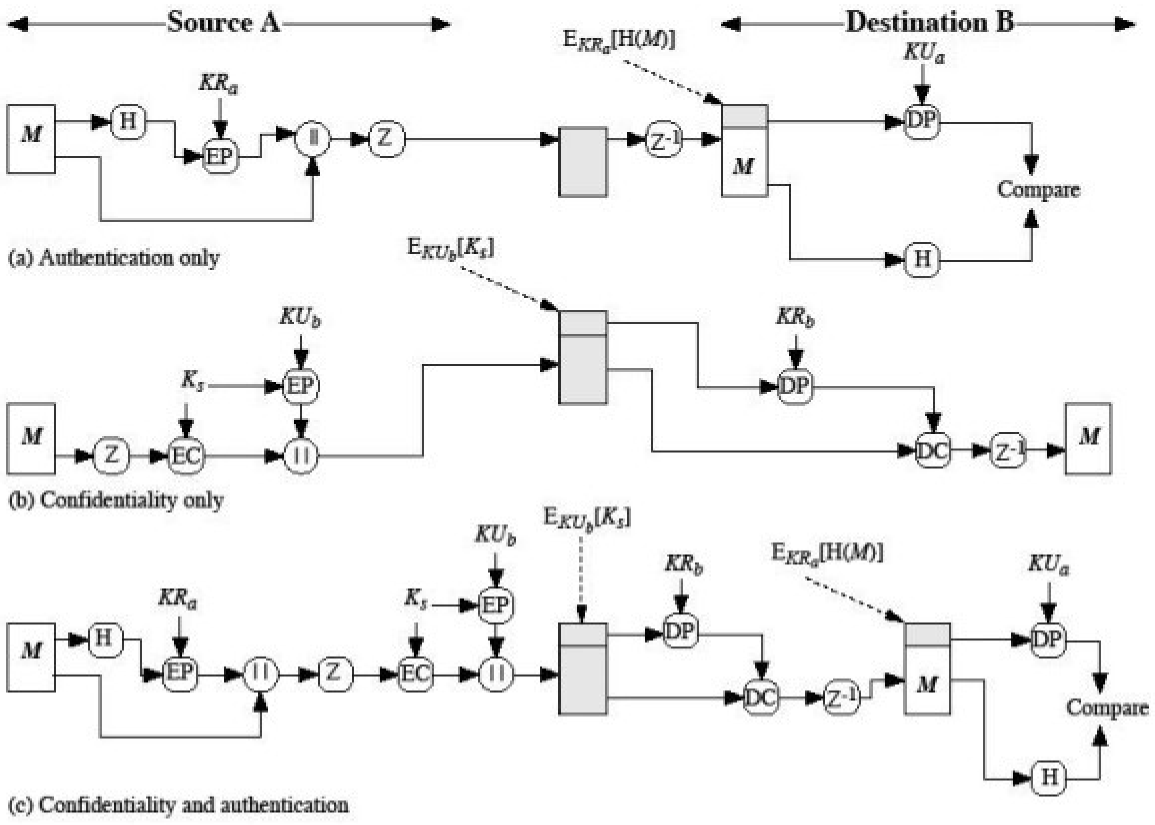


Fig.4.5.1.1:PGP Cryptographic Functions

Here both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext plus the signature is encrypted using CAST-128 and the session key is encrypted using RSA.

3. Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space for both e-mail transmission and for file storage.

The signature is generated before compression for two reasons:

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

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

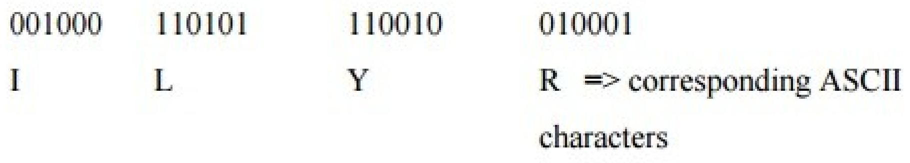
Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult. The compression algorithm used is ZIP.

4. e-mail compatibility

Many electronic mail systems only permit the use of blocks consisting of ASCII texts. To accommodate this restriction, PGP provides the service of

converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is **radix-64 conversion**. Each group of three octets of binary data is mapped into four ASCII characters.

e.g., consider the 24-bit (3 octets) raw text sequence 00100011 01011100 10010001, we can express this input in block of 6-bits to produce 4 ASCII characters.



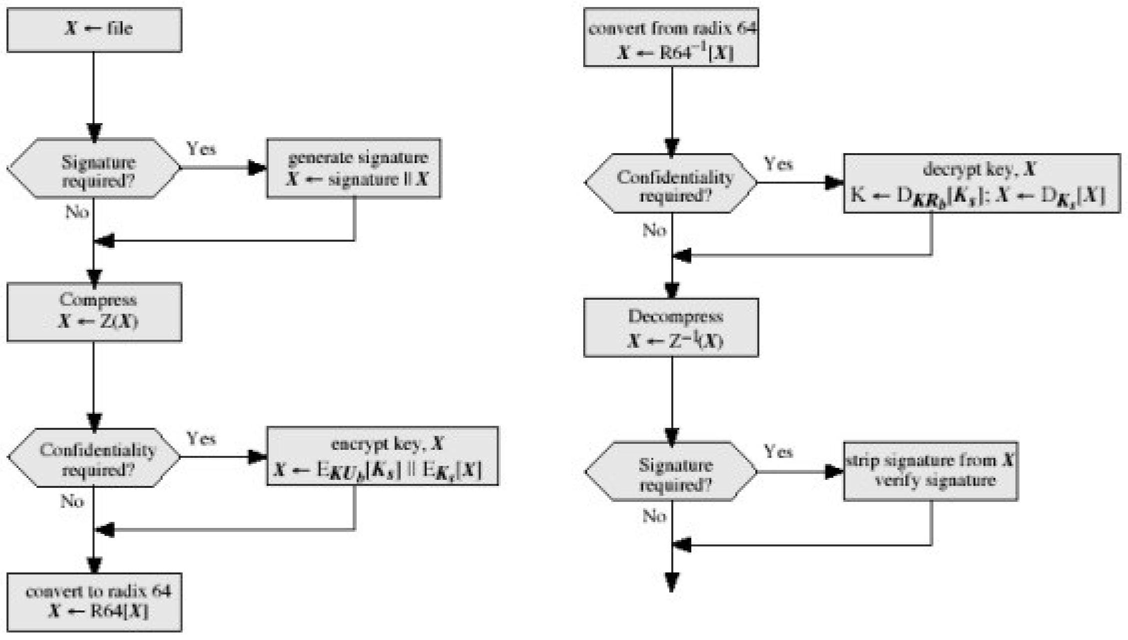
001000	110101	110010	010001
I	L	Y	R => corresponding ASCII characters

5. Segmentation and reassembly

E-mail facilities often are restricted to a maximum length. E.g., 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 the other processing, including the radix-64 conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the other steps.

2. PGP Operation Summary:



$X \leftarrow$ file
Signature required?
Yes
No
generate signature $X \leftarrow$ signature $\|$ X
Compress $X \leftarrow Z(X)$
Confidentiality required?
Yes
No
encrypt key, X $X \leftarrow E_{KU_b}[K_s] \| E_{K_s}[X]$
convert to radix 64 $X \leftarrow R64[X]$
convert from radix 64 $X \leftarrow R64^{-1}[X]$
Confidentiality required?
Yes
No
decrypt key, X $K \leftarrow D_{KR_b}[K_s]$; $X \leftarrow D_{K_s}[X]$
Decompress $X \leftarrow Z^{-1}(X)$
Signature required?
Yes
No
strip signature from X verify signature

(a) Generic Transmission Diagram (from A) (b) Generic Reception Diagram (to B)

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

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

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.

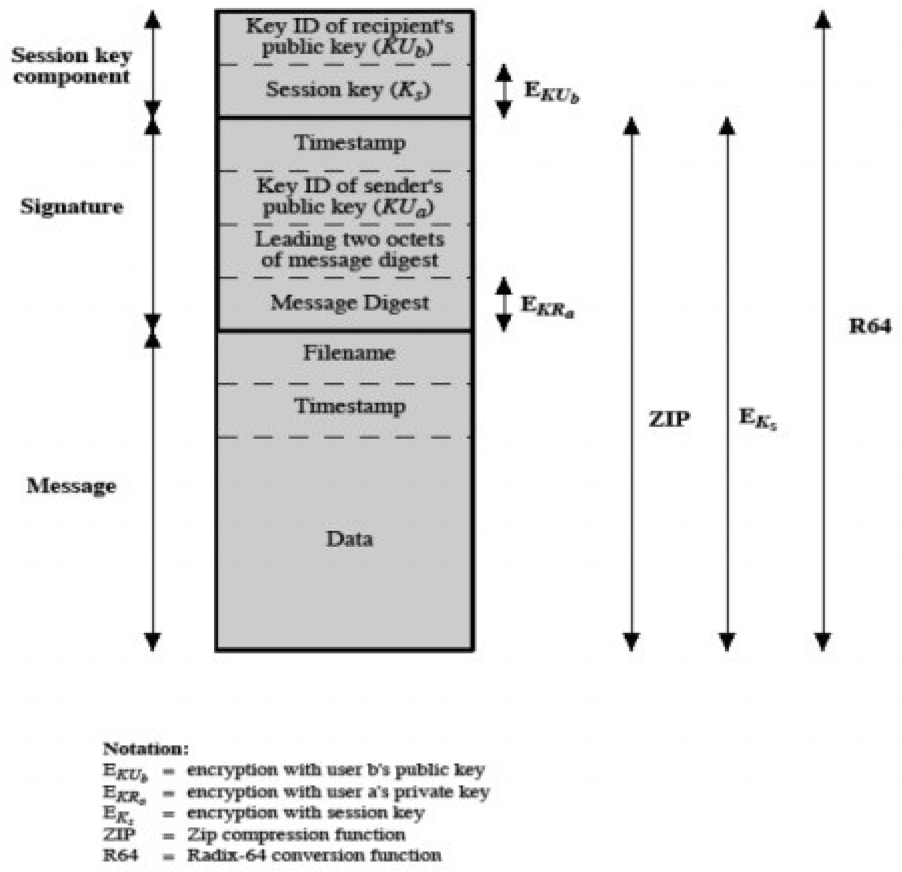
Signature component – includes the following

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.


Session key component
Key ID of recipient's public key (KU_b)
Session key (K_s)
E_{KU_b}
Signature
Timestamp
Key ID of sender's public key (KU_a)
Leading two octets of message digest
Message Digest
E_{KR_a}
Message
Filename
Timestamp
Data
ZIP
E_{K_s}
R64
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

3. 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.

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

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

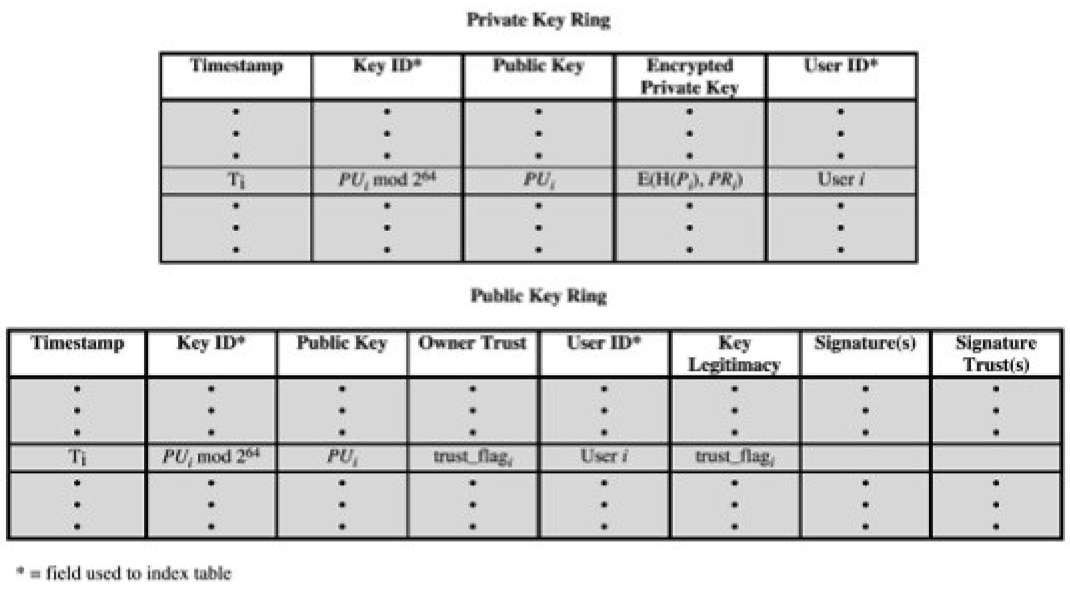
Key ID – the least significant bits of the public key.

Public key – public key portion of the pair.

Private key – private key portion of the pair.

User ID – the owner of the key.

Key legitimacy field – indicates the extent to which PGP will trust that this is a valid public key for this user.



Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
Ti	PU_i 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)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ti	PU_i mod 2^{64}	PU_i	$trust\_flag_i$	User *i*	$trust\_flag_i$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

* = field used to index table

Fig.4.5.3.1 General Structure of Private and Public Rings

Fig.4.5.3.1 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:

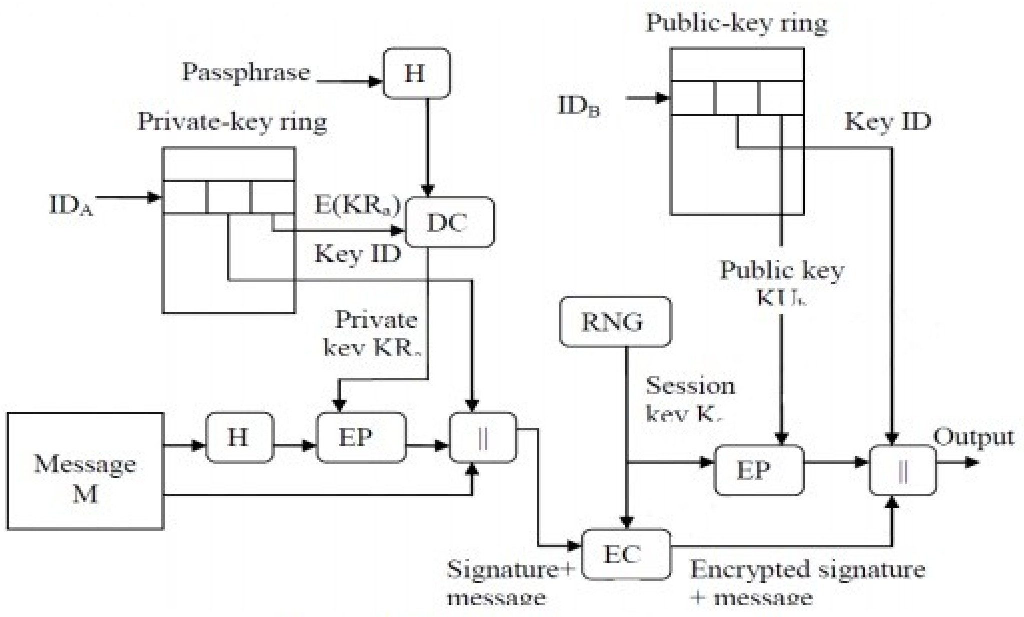


Figure 4.5.3.1: PGP message generation

1. 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 passpharse (password) to recover the unencrypted private key.

The signature component of the message is constructed.

2. 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.

The receiving PGP entity performs the following steps

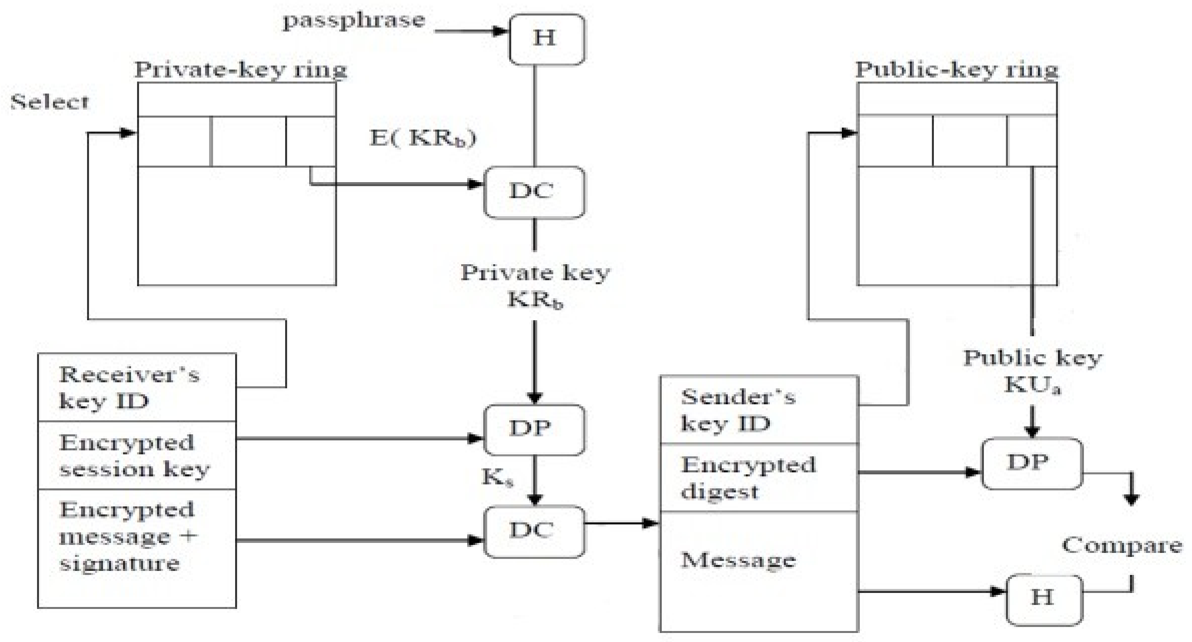


Figure: PGP message reception

1. 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 passpharse (password) to recover the unencrypted private key.

PGP then recovers the session key and decrypts the message.

2. 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 e-mail format standard, based on technology from RSA Data Security. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851.

1. 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. Following are the limitations of SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects.

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 is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations. The specification is provided in RFCs 2045 through 2049.

2. Overview

The MIME specification includes the following elements:

1. **Five new message header** fields are defined, which may be included in an RFC 822 header. These fields provide information about the body of the message.

2. **A number of content formats** are defined, thus standardizing representations that support multimedia electronic mail.

3. **Transfer encodings** are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

   In this subsection, we introduce the five message header fields. The next two subsections deal with content formats and transfer encodings.

3. The five header fields defined in MIME are as follows:

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

Content-Type: Describes the data contained in the body with sufficient detail

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

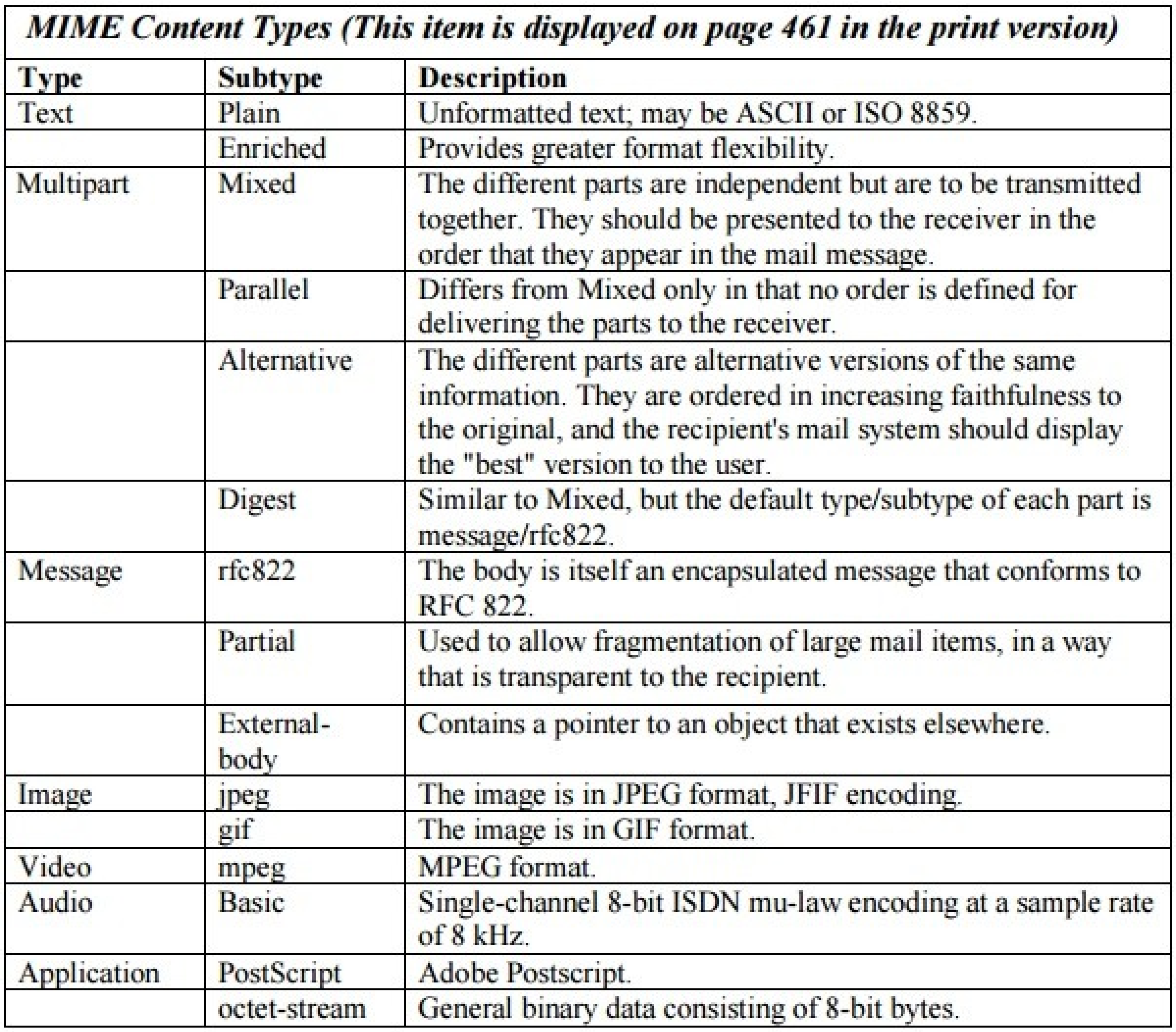
Content-ID: Used to identify MIME entities uniquely in multiple contexts.

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

4. MIME Content Types

The bulk of the MIME specification is concerned with the definition of a variety of content types. This reflects the need to provide standardized ways of dealing with a wide variety of information representations in a multimedia environment.

Below lists the content types specified in RFC 2046. There are seven different major types of content and a total of 15 subtypes



MIME Content Types (This item is displayed on page 461 in the print version)

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

For the text type of body, no special software is required to get the full meaning of the text, aside from support of the indicated character set. The primary subtype is plain text, which is simply a string of ASCII characters or ISO 8859 characters. The enriched subtype allows greater formatting flexibility. The multipart type indicates that the body contains multiple, independent parts. The Content-Type header field includes a parameter, called boundary, that defines the delimiter between body parts.

The multipart/digest subtype is used when each of the body parts is interpreted as an RFC 822 message with headers. This subtype enables the construction of a message whose parts are individual messages. For example, the moderator of a group might collect e-mail messages from participants, bundle these messages, and send them out in one encapsulating MIME message.

The message type provides a number of important capabilities in MIME. The message/rfc822 subtype indicates that the body is an entire message, including header and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 822 message, but also any MIME message.

The message/partial subtype enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this subtype, three parameters are specified in the Content-Type: Message/Partial field: an id common to all fragments of the same message, a sequence number unique to each fragment, and the total number of fragments.

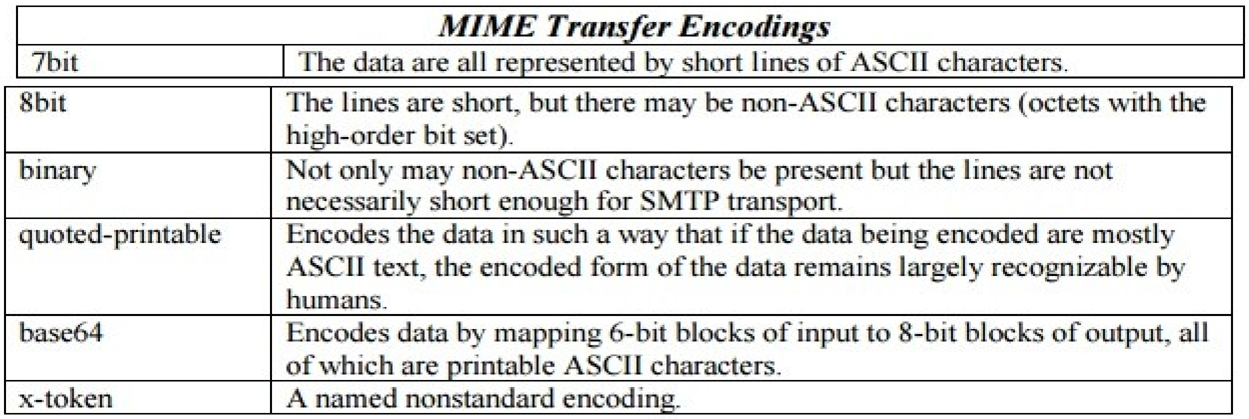
The message/external-body subtype indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. As with the other message types, the message/external-body subtype has an outer header and an encapsulated message with its own header. The only necessary field in the outer header is the Content-Type field, which identifies this as a message/external-body subtype. The inner header is the message header for the encapsulated message. The Content-Type field in the outer header must include an access-type parameter, which indicates the method of access, such as FTP (file transfer protocol).

The application type refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-based application.

5. MIME Transfer Encodings

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

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values. For SMTP transfer, it is safe to use the 7bit form. The 8bit and binary forms may be usable in other mai transport contexts. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used, for which a name is to be supplied. The two actual encoding schemes defined are quoted-printable and base64.



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

The quoted-printable transfer encoding is useful when the data consists largely of octets that correspond to printable ASCII characters. In essence, it represents nonsafe characters by the hexadecimal representation of their code and introduces reversible (soft) line breaks to limit message lines to 76 characters.

The base64 transfer encoding, also known as radix-64 encoding, is a common one for encoding arbitrary binary data in such a way as to be invulnerable to the processing by mail transport programs.

Canonical Form

An important concept in MIME and S/MIME is that of canonical form. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in contrast to native form, which is a format that may be peculiar to a particular system.

UNIT - IV

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 capabil- ity 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 sec ure 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 authentica- tion and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic co mmerce applications have builtin security protocols, the use of IPsec enhancesthat security. IPsec g uarantees that all traffic designated by the network administrator is both encrypted and authenticate d, 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 applica- tions is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed appli cations (including remote logon, client/server, e-ail, file transfer, Web access, and so on) can be secured.

Figure 19.1 is a typical scenario of IPsec usage. An organization maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPsec protocols are used. These protocols operate in networking devices, such as a router or firewall, that con- nect each LAN to the outside world. The IPsec networking device will typically encrypt and compress all traffic going into the WAN and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPsec protocols to provide security.

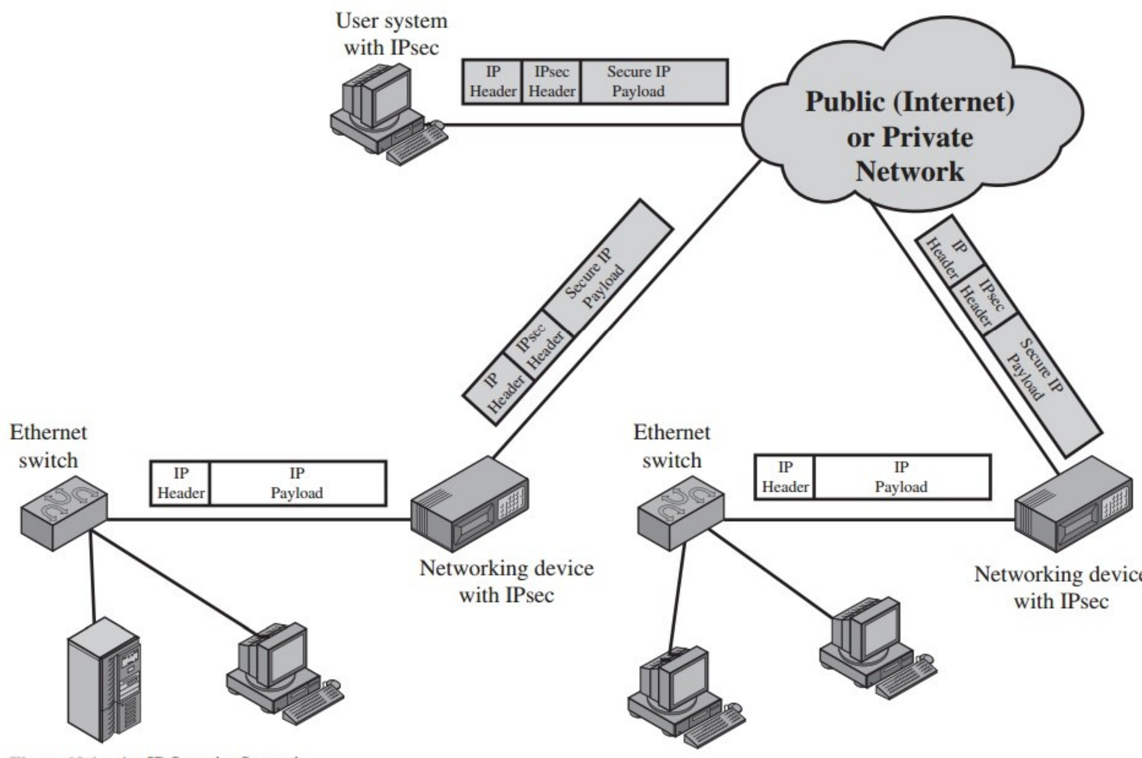


Figure 19.1 An IP Security Scenario

Benefits of IPsec

Some of the 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 mech anisms, 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.

Routing Applications

In addition to supporting end users and protecting premises systems and networks, IPsec can play a vital role in the routing architecture required for internetworking. [HUIT98] lists the following examples of the use of IPsec. IPsec can assure that

- A router advertisement (a new router advertises its presence) comes from an authorized router.
- A neighbor advertisement (a router seeks to establish or maintain a neighbor relationship with a router in another routing domain) comes from an autho- rized router.
- A redirect message comes from the router to which the initial IP packet was sent.
- A routing update is not forged.

Without such security measures, an opponent can disrupt communications or divert some traffic. Routing protocols such as Open Shortest Path First (OSPF) should be run on top of security associations between routers that are defined by IPsec.

IPsec Documents

IPsec encompasses three functional areas: authentication, confidentiality, and key management. The totality of the IPsec specification is scattered across dozens of

RFCs and draft IETF documents, making this the most complex and difficult to grasp of all IETF specifications. The best way to grasp the scope of IPsec is to consult the latest version of the IPsec document roadmap, which as of this writing is [FRAN09]. The docum ents can be categorized into the following groups.

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPsec technology. The current specification is RFC 4301, *Security Architecture for the Internet Protocol*.
- **Authentication Header (AH):** AH is an extension header to provide message authentication. The current specification is RFC 4302, *IP Authentication Header*. Because message authentication is provid ed by ESP, the use of AH is deprecated. It is included in IPsecv3 for backward compatibility but should not be used in new applications. We do not discuss AH in this chapter.
- **Encapsulating Security Payload (ESP):** ESP consists of an encapsulating head er and trailer used to provide encryption or combined encryption/authentication. The current specification is RFC 4303, *IP Encapsulating Security Paylo ad (ESP)*.
- **Internet Key Exchange (IKE):** This is a collection of documents describing the key management schemes for use with IPsec. The main specification is RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, but there are a number of related RFCs.
- **Cryptographic algorithms:** This category encompasses a large set of documents that define and describe cryptographic algorithms for encryption, message authentication, pseudorandom functions (PRFs), and cryptographic key exchange.
- **Other:** There are a variety of other IPsec-related RFCs, including those deal-ing with security policy and management information base (MIB) content.

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). RFC 4301 lists the following services:

- Access control
- Connectionless integrity
- Data origin authentication

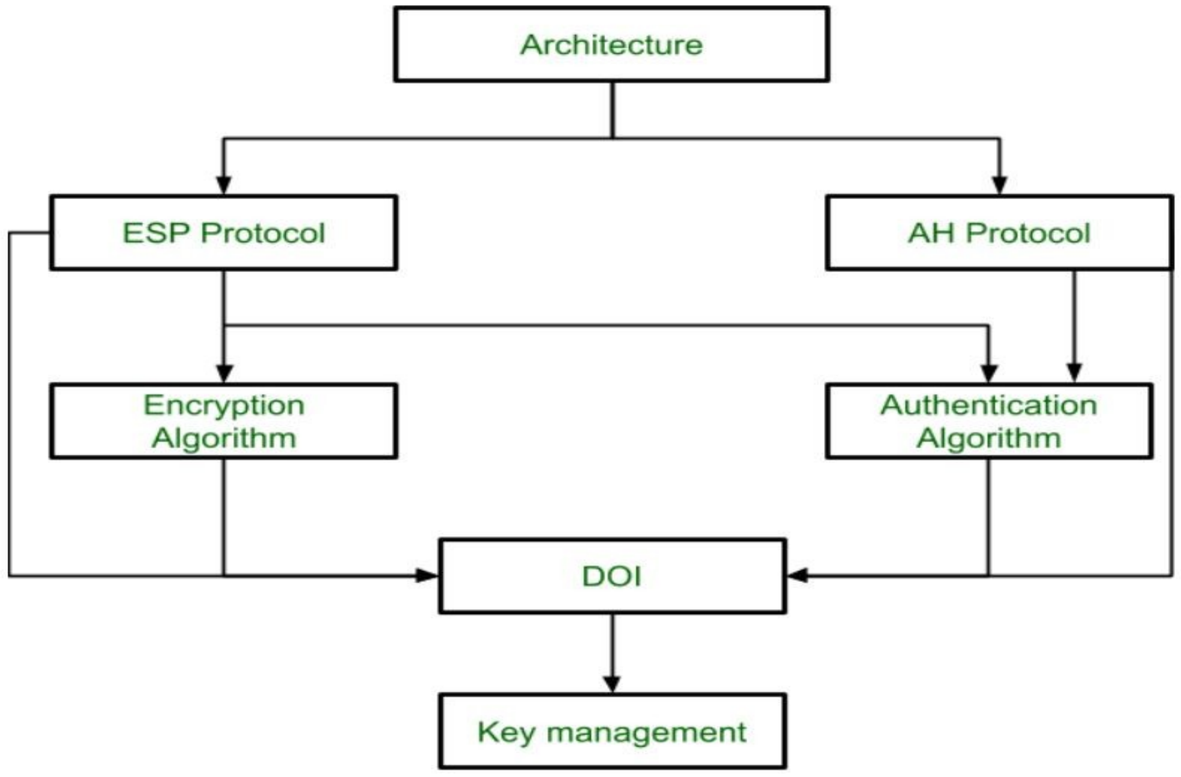
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

IPSec Architecture

IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture include protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integirity

IP Security Architecture:

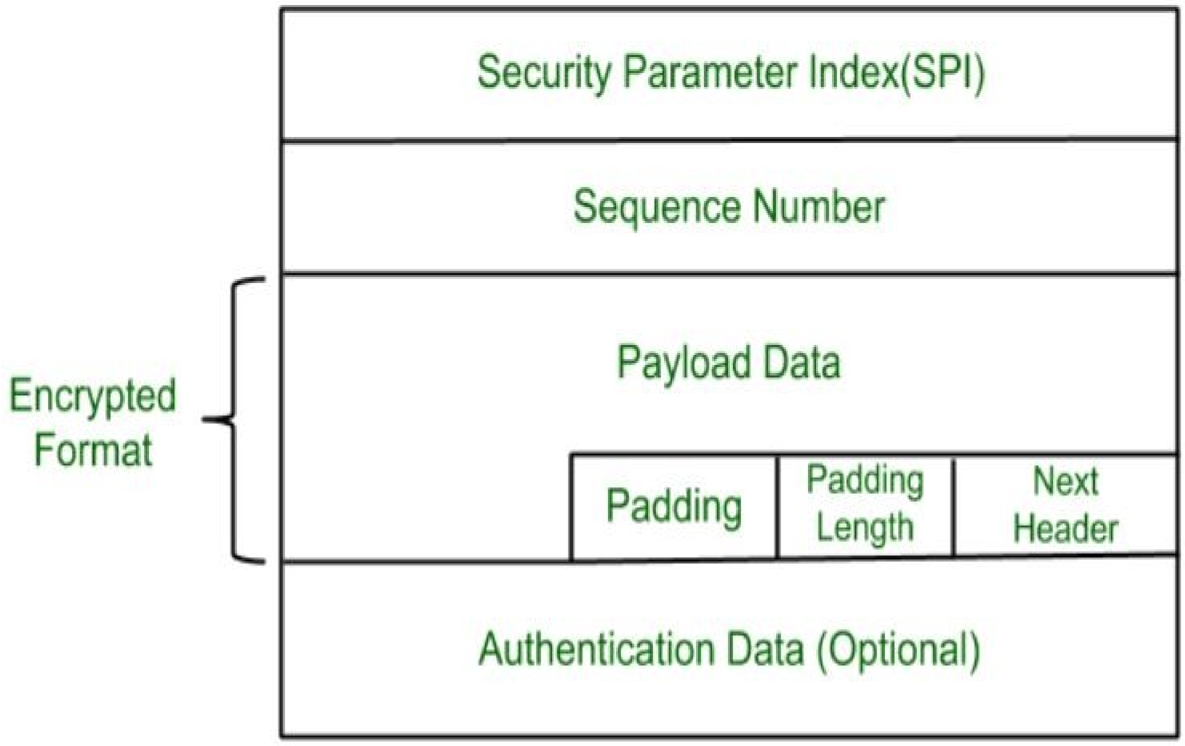


1. Architecture:
Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.

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.

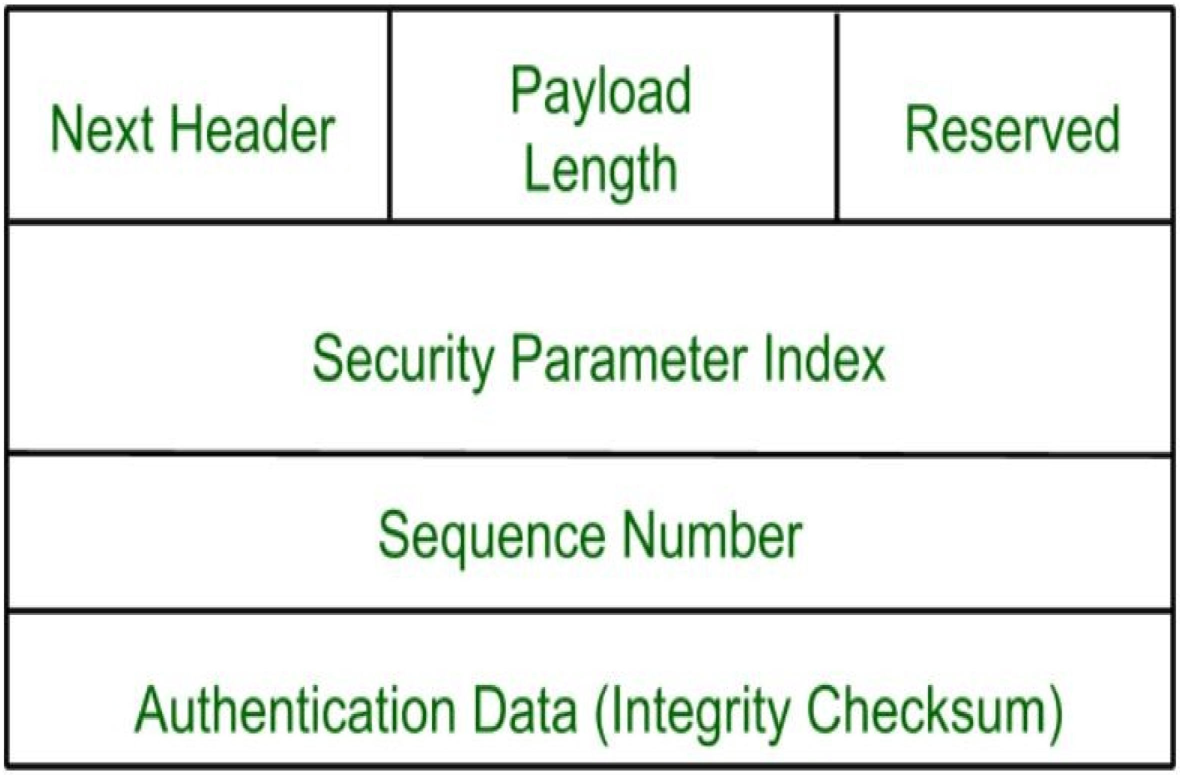
Packet Format:



- **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.

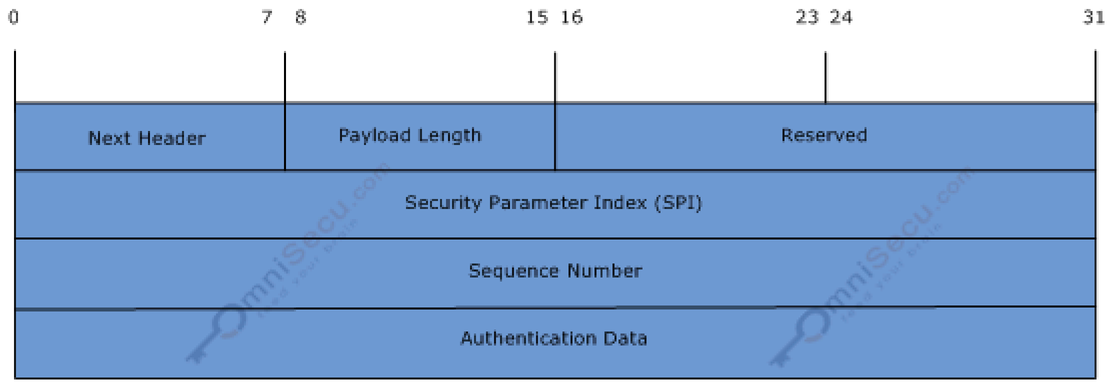


Figure: Authentication Header (AH) - Header

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.

ENCAPSULATING SECURITY PAYLOAD

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

ESP Format:

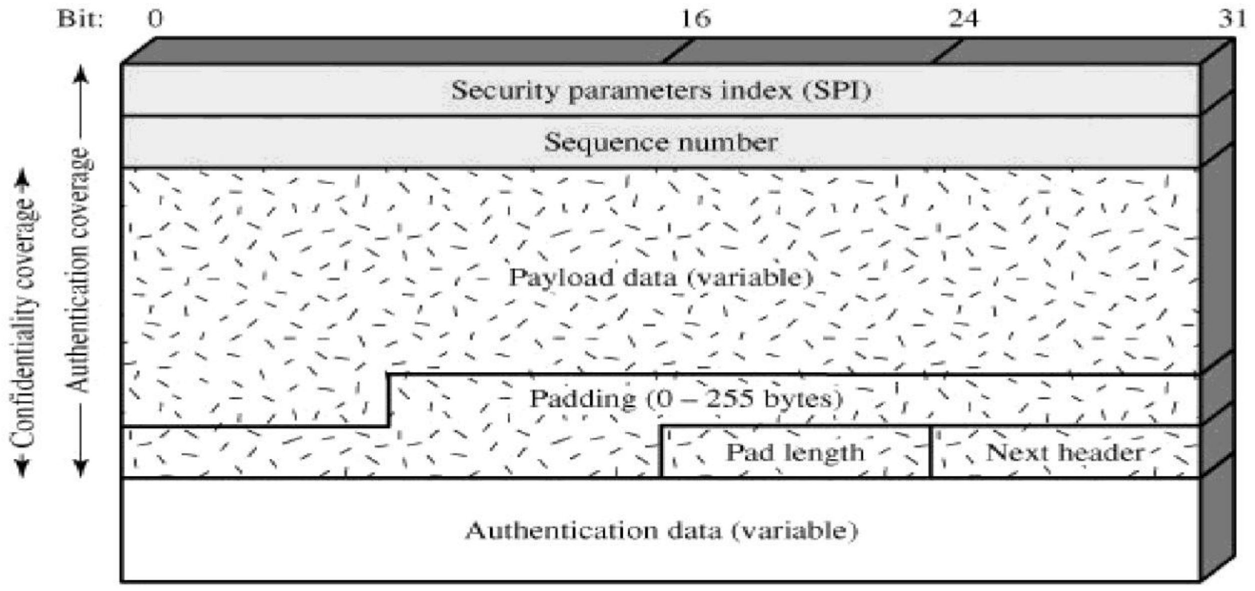


Figure 1.7. IPSec ESP format

Figure 1.7 shows the format of an ESP packet. It contains the following fields:
• Security Parameters Index (32 bits): Identifies a security association.
• Sequence Number (32 bits): A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
• Payload Data (variable): This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
• Padding (0255 bytes): The purpose of this field is discussed later.
• Pad Length (8 bits): Indicates the number of pad bytes immediately preceding this field.
• Next Header (8 bits): Identifies the type of data contained in the payload data field by identifying the first header in that payload
• Authentication Data (variable): A variable-length field (must be an integral number of 32-bit words) that contains the Integrity. Check Value computed over the ESP packet minus the Authentication Data field.

Encryption and Authentication Algorithms: The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESP service. If the algorithm used to encrypt the payload requires cryptographic synchronization data, such as an initialization vector (IV), then these data may be carried explicitly at the beginning of the Payload Data field. If included, an IV is usually not encrypted, although it is often referred to as being part of the ciphertext. The current specification dictates that a compliant implementation must support DES in cipher block chaining (CBC) mode. A number of other algorithms have been assigned identifiers in the DOI document and could therefore easily be used for encryption; these include

- Three-key triple DES
- RC5
- IDEA
- Three-key triple IDEA
- CAST
- Blowfish

As with AH, ESP supports the use of a MAC with a default length of 96 bits. Also as with AH, the current specification dictates that a compliant implementation must support HMAC- MD5-96 and HMAC-SHA-1-96.

Padding: The Padding field serves several purposes:

- If an encryption algorithm requires the plaintext to be a multiple of some number of bytes (e.g., the multiple of a single block for a block cipher), the Padding field is used to expand the plaintext (consisting of the Payload Data, Padding, Pad Length, and Next Header fields) to the required length.
- The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.
- Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

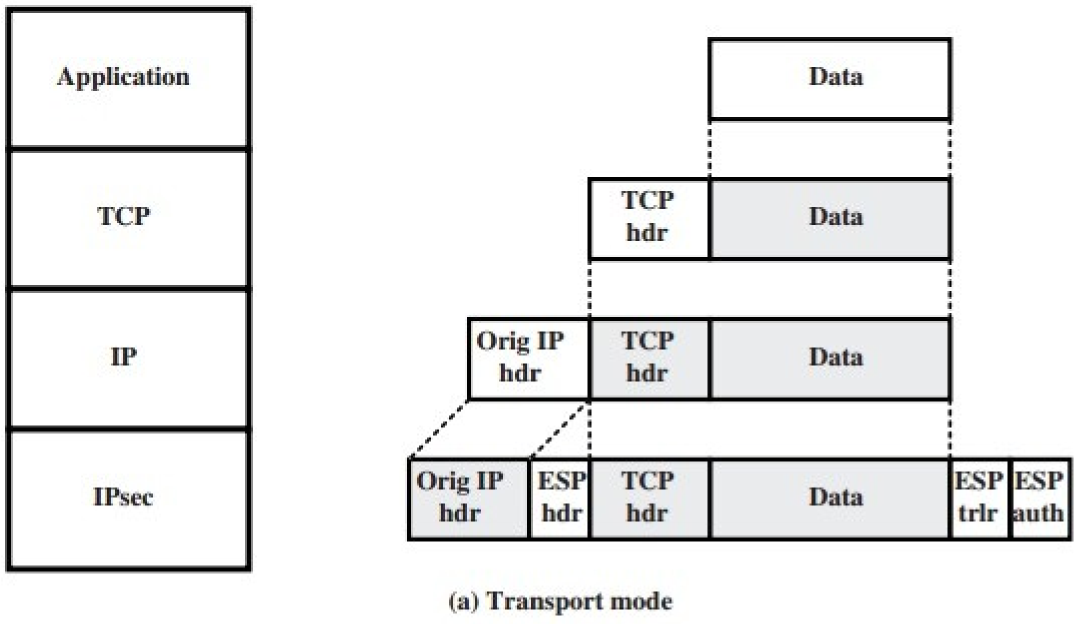
COMBINING SECURITY ASSOCIATIONS

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 fire-walls. 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.



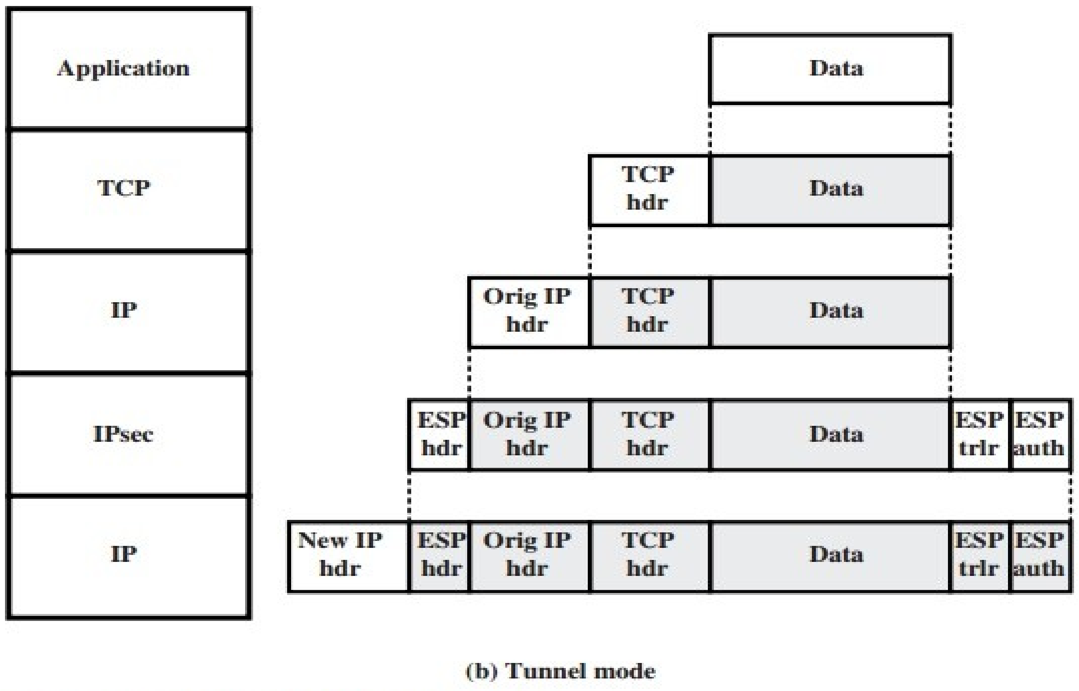


Figure 19.9 Protocol Operation for ESP

The two approaches can be combined, for example, by having a transport SA between hosts t ravel part of the way through a tunnel SA between security gateways. One interesting issue that arises when considering SA bundles is 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.

Authentication Plus Confidentiality

Encryption and authentication can be combined in order to transmit an IP packet that has both confidentiality and authentication between hosts. We look at several approaches.

ESP WITH AUTHENTICATION OPTION This approach is illustrated in Figure 19.8. In this approach, the user first applies ESP to the data to be protected and then appends the authentication data field. There are actually two subcases:

• **Transport mode ESP:** Authentication and encryption apply to the IP payload delivered to the host, but the IP header is not protected.

• **Tunnel mode ESP:** Authentication applies to the entire IP packet delivered to the outer IP destination address (e.g., a firewall), and authentication is performed at that destination. The entire inner IP packet is protected by the privacy mechanism for delivery to the inner IP destination.

For both cases, authentication applies to the ciphertext rather than the plaintext.

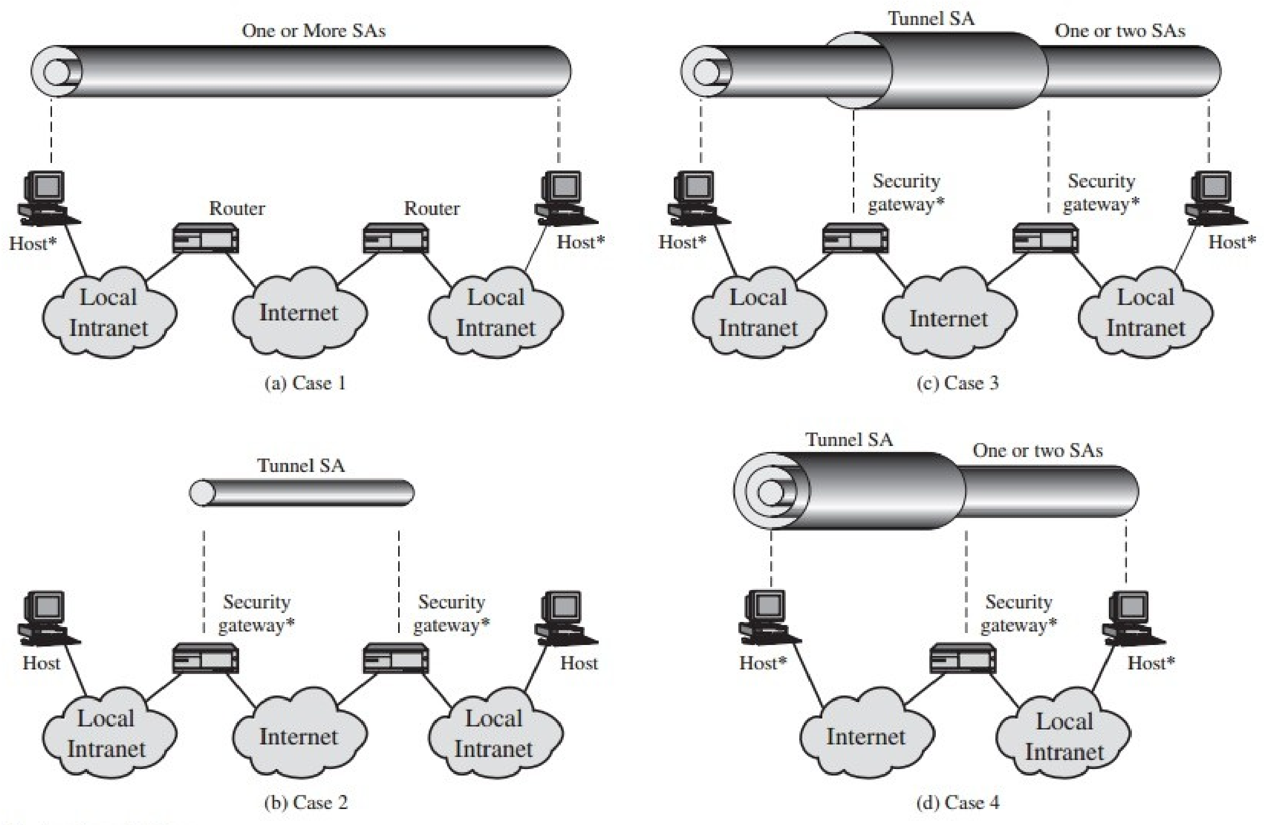
TRANSPORT ADJACENCY Another way to apply authentication after encryption is to use two bundled transport SAs, with the inner being an ESP SA and the outer being
an AH SA. In this case, ESP is used without its authentication option. Because the inner SA is a transport SA, encryption is applied to the IP payload. The resulting packet consists of an IP header (and possibly IPv6 header extensions) followed by an ESP. AH is then applied in transport mode, so that authentication covers the ESP plus
the original IP header (and extensions) except for mutable fields. The advantage of
this approach over simply using a single ESP SA with the ESP authentication option is that the a uthentication covers more fields, including the source and destination IP addresses. The disadvant age is the overhead of two SAs versus one SA.

TRANSPORT-TUNNEL BUNDLE The use of authentication prior to encryption might be preferable for several reasons. First, because the authentication data are protected by encryption, it is impossible for anyone to intercept the message and alter the authentication data without detection. Second, it may be desirable to store the authentic ation information
with the message at the destination for later reference. It is more convenient to do this if the aut hentication information applies to the unencrypted message; otherwise the message would have to be reencrypted to verify the authentication information.

One approach to applying authentication before encryption between two hosts is to use a bundle consisting of an inner AH transport SA and an outer ESP tunnel SA. In this case, authentication is applied to the IP payload plus the IP header (and extensions) except for mutable fields. The resulting IP packet is then processed in tunnel mode by ESP; the result is that the entire, authenticated inner packet is encrypted and a new outer IP header (and extensions) is added.

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. The lower part



* = implements IPsec

Figure 19.10 Basic Combinations of Security Associations

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 appropri-
ate secret keys. Among the possible combinations are

a. AH in transport mode

b. ESP in transport mode

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

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

We have already discussed how these various combinations can be used to support authentication, encryption, authentication before encryption, and authenti- cation after encryption.

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 authenti- cation 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 combi- nations 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 ser- vices 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 fir ewall 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.

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 [GARF02], 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

Table 16.1 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 secu- rity; Part Four of this book addresses the issue of system security in general but is also applicable to Web system security. Issues of traffic security fall into the category of network security and are addressed in this chapter.

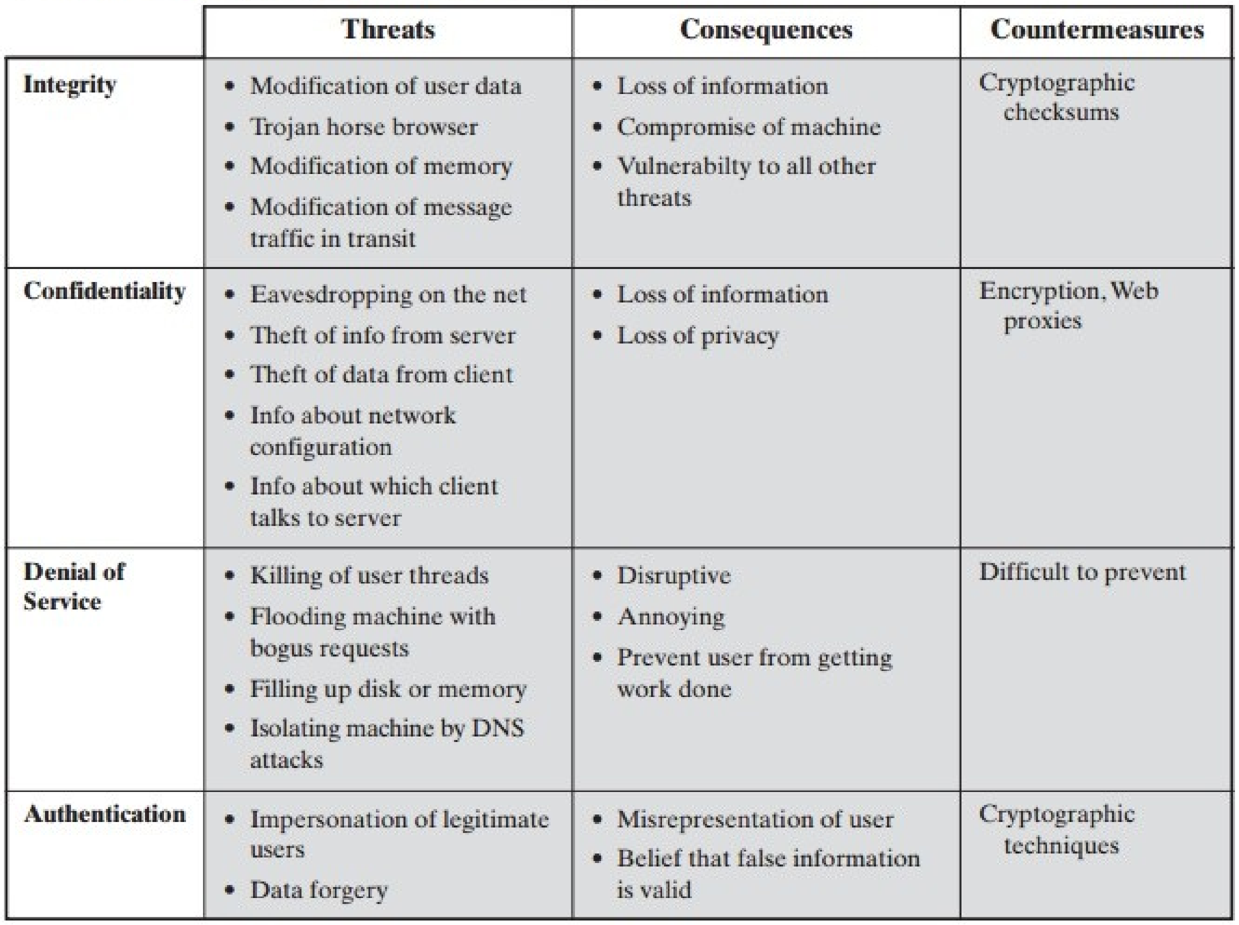
Web Traffic Security Approaches

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.

Figure 16.1 illustrates this difference. One way to provide Web security is to use IP security (IPsec) (Figure 16.1a). The advantage of using IPsec is that it is trans- parent to e nd 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.

Another relatively general-purpose solution is to implement security just above TCP (Figure 16.1b). The foremost example of this approach is the Secure

Table 16.1 A Comparison of Threats on the Web



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

Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS). At this level, there are two implementation choices. For full general- ity, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

Application-specific security services are embedded within the particular appli- cation. Figure 16.1c shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.

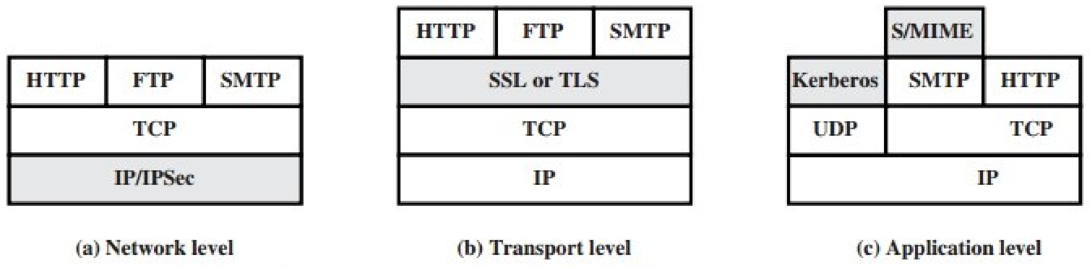


Figure 16.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

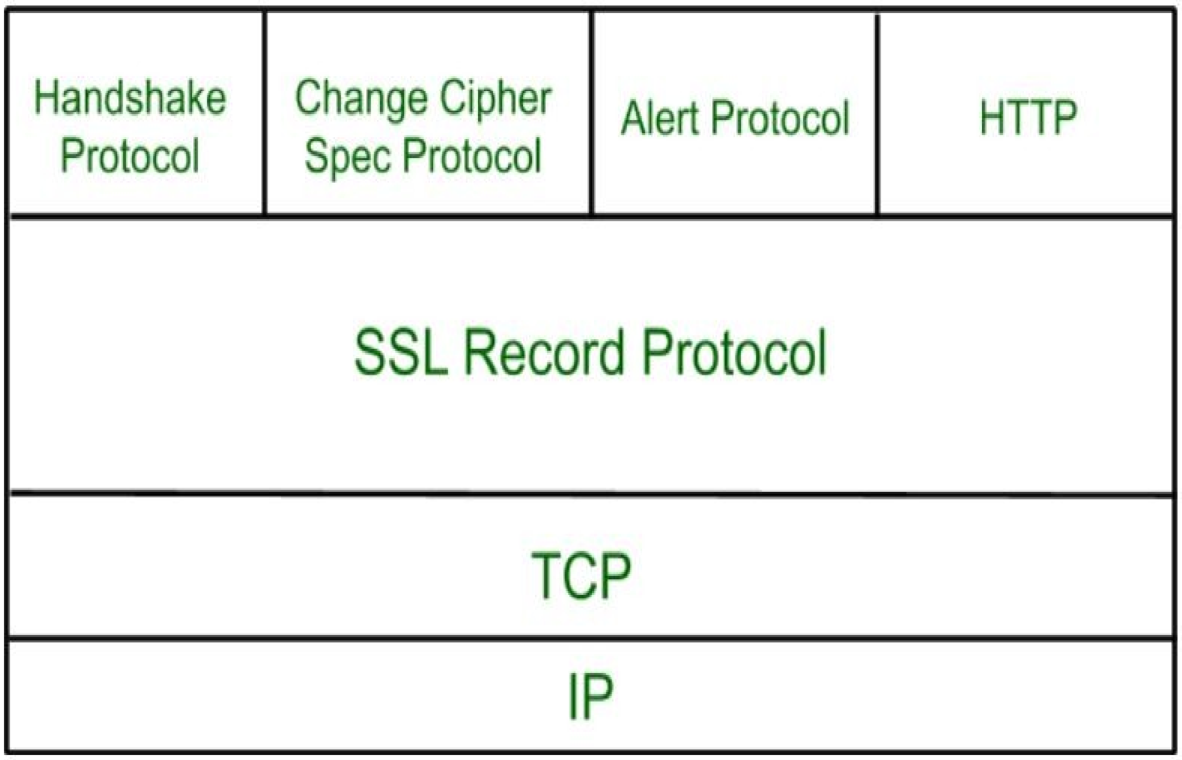
Secure Socket Layer (SSL)

It provides security to the data that is transferred between web browser and server. SSL encrypt the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

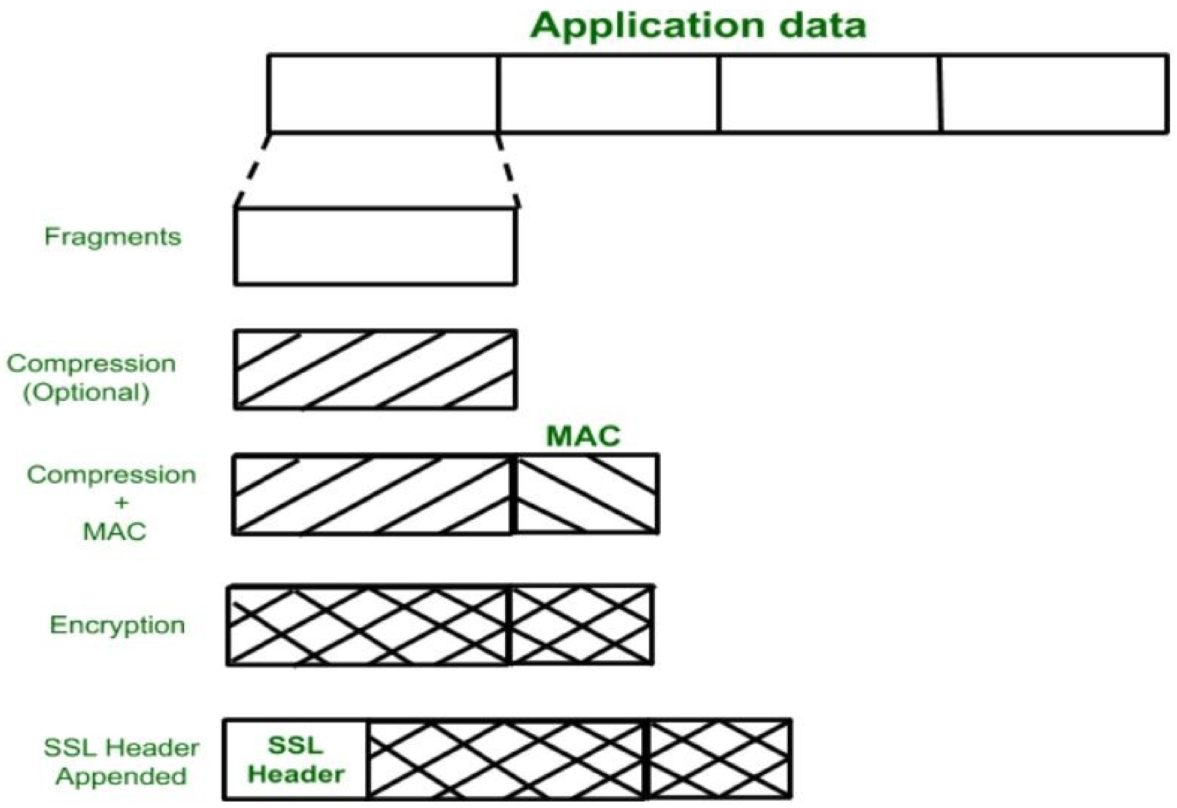
SSL Protocol Stack:



SSL Record Protocol:

SSL Record provide two services to SSL connection.
- Confidentiality
- Message Integerity

In SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data.



Handshake Protocol:

Handshake Protocol is used to establish sessions. This protocol allow client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.
- **Phase-1:** In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purpose.
- **Phase-2:** Server send his certificate and Server-key-exchange. Server end the phase-2 by sending Server-hello-end packet.
- **Phase-3:** In this phase Client reply to the server by sending his certificate and Client-exchange-key.
- **Phase-4:** In Phase-4 Change-cipher suite occurred and after this Handshake Protocol ends.

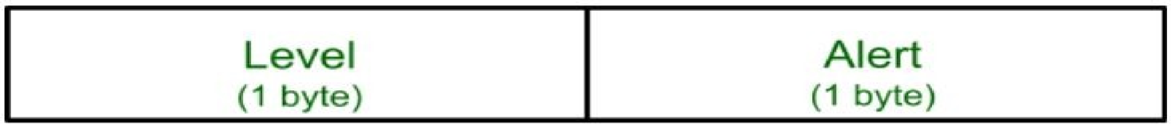
Change-cipher Protocol:

This protocol uses SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in pending state. After handshake protocol the Pending state is converted into Current state.

Change-cipher protocol consists of single message which is 1 byte in length and can have only one value. This protocol purpose is to cause the pending state to be copied into current state.

Alert Protocol:

This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contain 2 bytes.



Level is further classified into two parts:

- **Warning:**
  This Alert have no impact on the connection between sender and receiver.
- **Fatal Error:**
  This Alert breaks the connection between sender and receiver.

Silent Features of Secure Socket Layer:

- Advantage of this approach is that the service can be tailored to the specific needs of the given application.
- Secure Socket Layer was originated by Netscape.
- SSL is designed to make use of TCP to provide reliable end-to-end secure service.
- This is two-layered protocol.

Transport Layer Security (TLS)

Transport Layer Securities (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 eavdrops 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 is 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.

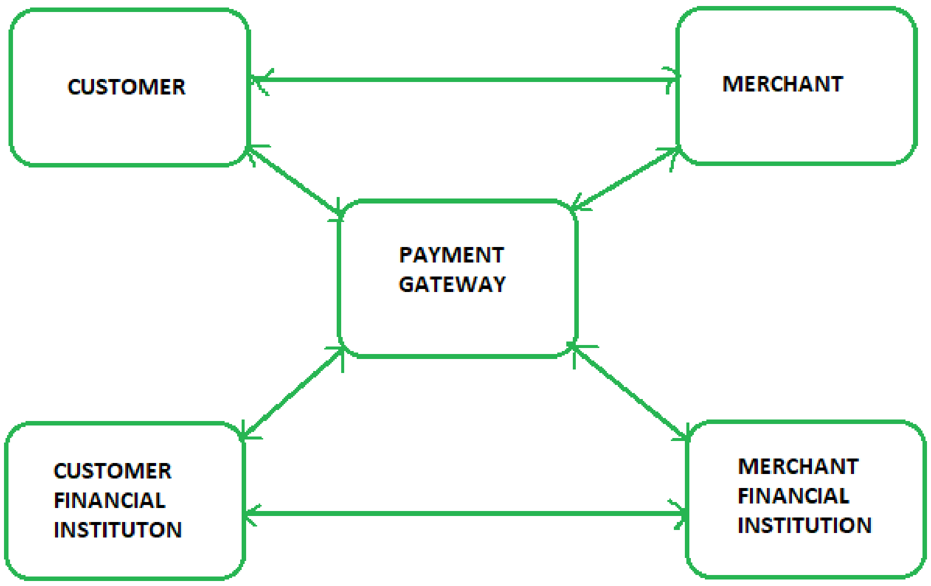
There are some differences between SSL and TLS which are given below:

S.NO	SSL	TLS
1.	SSL stands for Secure Socket Layer.	TLS stands for Transport Layer Security.
2.	SSL (Secure Socket Layer) supports **Fortezza** algorithm.	TLS (Transport Layer Security) does not supports **Fortezza** algorithm.
3.	SSL (Secure Socket Layer) is the 3.0 version.	TLS (Transport Layer Security) is the 1.0 version.
4.	In SSL(Secure Socket Layer), Message digest is used to create master secret.	In TLS(Transport Layer Security), Pseudo-random function is used to create master secret.
5.	In SSL(Secure Socket Layer), Message Authentication Code protocol is used.	In TLS(Transport Layer Security), Hashed Message Authentication Code protocol is used.
6.	SSL (Secure Socket Layer) is complex than TLS(Transport Layer Security).	TLS (Transport Layer Security) is simple.

Secure Electronic Transaction (SET) Protocol

Secure Electronic Transaction or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT) and NetScape which provided technology of Secure Socket Layer (SSL).
SET protocol restricts revealing of credit card details to merchants thus keeping hackers and thieves at bay. SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

Before discussing SET further, let's see a general scenario of electronic transaction, which includes client, payment gateway, client financial institution, merchant and merchant financial institution.



Requirements in SET :

SET protocol has some requirements to meet, some of the important requirements are :

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.
- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of best security mechanisms.

Participants in SET :

In the general scenario of online transaction, SET includes similar participants:

1. **Cardholder** – customer
2. **Issuer** – customer financial institution
3. **Merchant**
4. **Acquirer** – Merchant financial
5. **Certificate authority** – Authority which follows certain standards and issues certificates(like X.509V3) to all other participants.

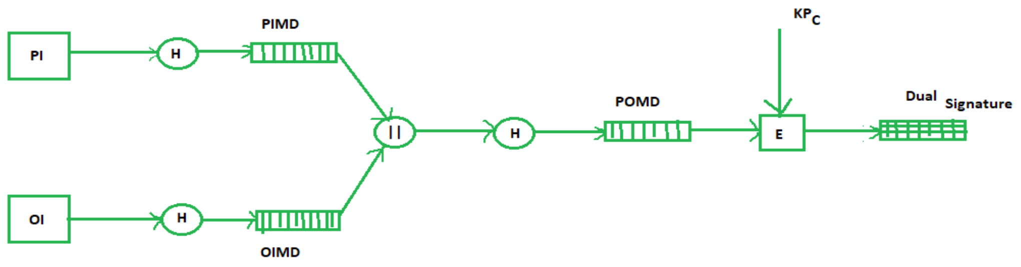
SET functionalities :

- **Provide Authentication**
  - **Merchant Authentication** – To prevent theft, SET allows customers to check previous relationships between merchant and financial institution. Standard X.509V3 certificates are used for this verification.
  - **Customer / Cardholder Authentication** – SET checks if use of credit card is done by an authorized user or not using X.509V3 certificates.
- **Provide Message Confidentiality** : Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purpose.
- **Provide Message Integrity** : SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1,

Dual Signature :

The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers :
Order Information (OI) for merchant
Payment Information (PI) for bank

You might think sending them separately is an easy and more secure way, but sending them in a connected form resolves any future dispute possible. Here is the generation of dual signature:



Where,

PI stands for payment information

OI stands for order information

PIMD stands for Payment Information Message Digest

OIMD stands for Order Information Message Digest

POMD stands for Payment Order Message Digest

H stands for Hashing

E stands for public key encryption

KPc is customer's private key

|| stands for append operation

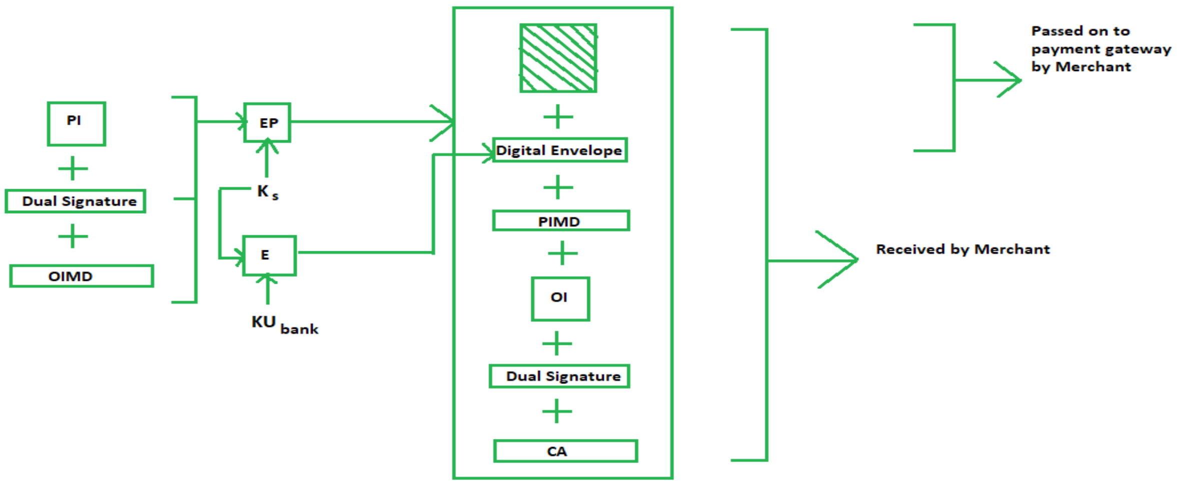
Dual signature, DS= E(KPc, [H(H(PI)||H(OI))])

Purchase Request Generation :

The process of purchase request generation requires three inputs:

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

The purchase request is generated as follows:



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

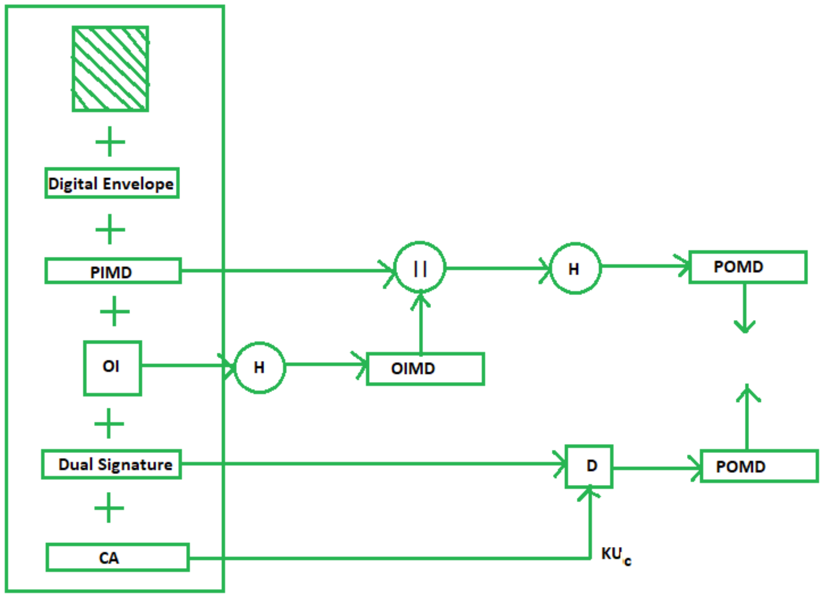
KUbank is public key of bank

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 KUc which is public key of customer or cardholder for decryption ‘D’.

Payment Authorization and Payment Capture :

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

7.1 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.

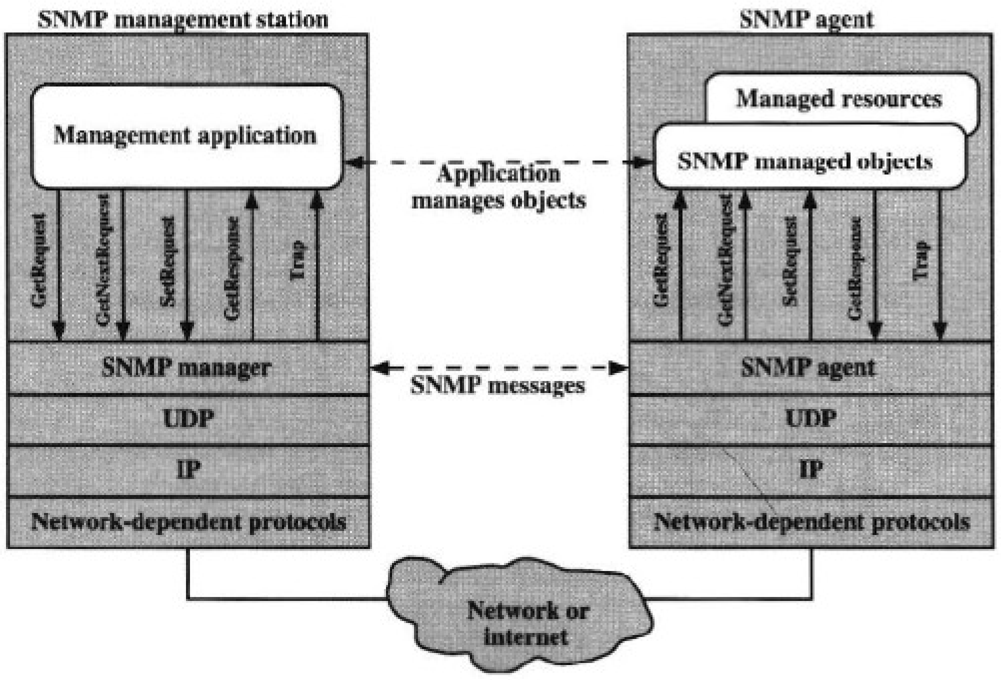
⇨ To manage resources in the network, each resource is represented as an object. An object is, essentially, a data variable that represents one aspect of the managed agent. The collection of objects is referred to as a **management information base (MIB).**

⇨ 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 received 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:

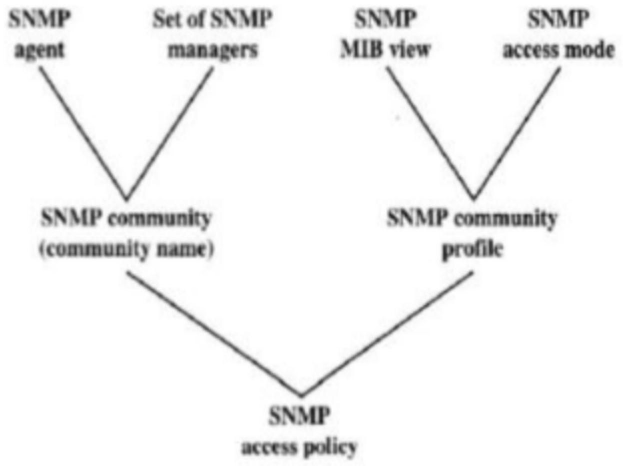
- **Authentication service:** The agent may wish to limit access to the MIB to authorized managers.
- **Access policy:** The agent may wish to give different access privileges to different managers.
- **Proxy service:** An agent may act as proxy to other agents. This may involve implementing the authentication 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 SNMPvl authentication service is to assure the recipient that an SNMPvl message is from the source that it c1aims to be from. SNMPvl only provides far a trivial scheme far authentication. Every message (get or put request) from a manager to an agent inc1udes 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-ONL Y, READ-WRITE}. An access mode is defined for each community.

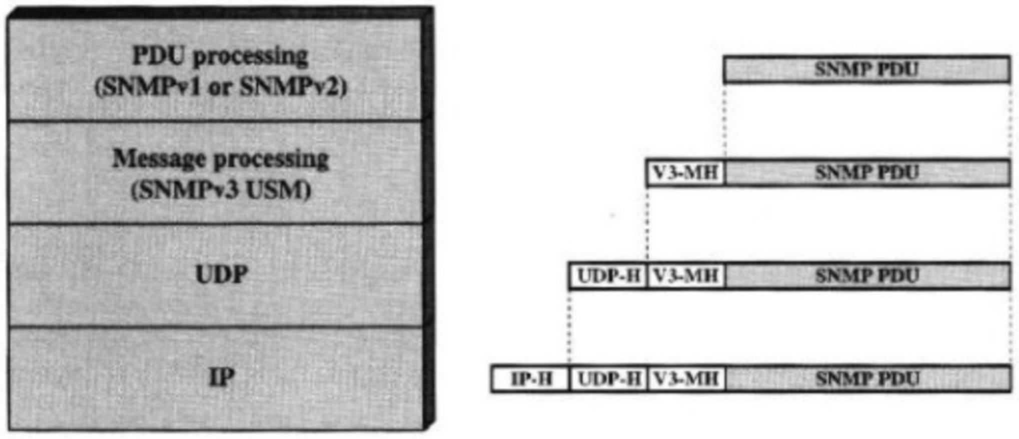


SNMPv1 Administrative Concepts

7.3 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.



IP-H = IP header
UDP-H = UDP header
V3-MH = SNMPv3 message header
PDU = Protocol data unit

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.

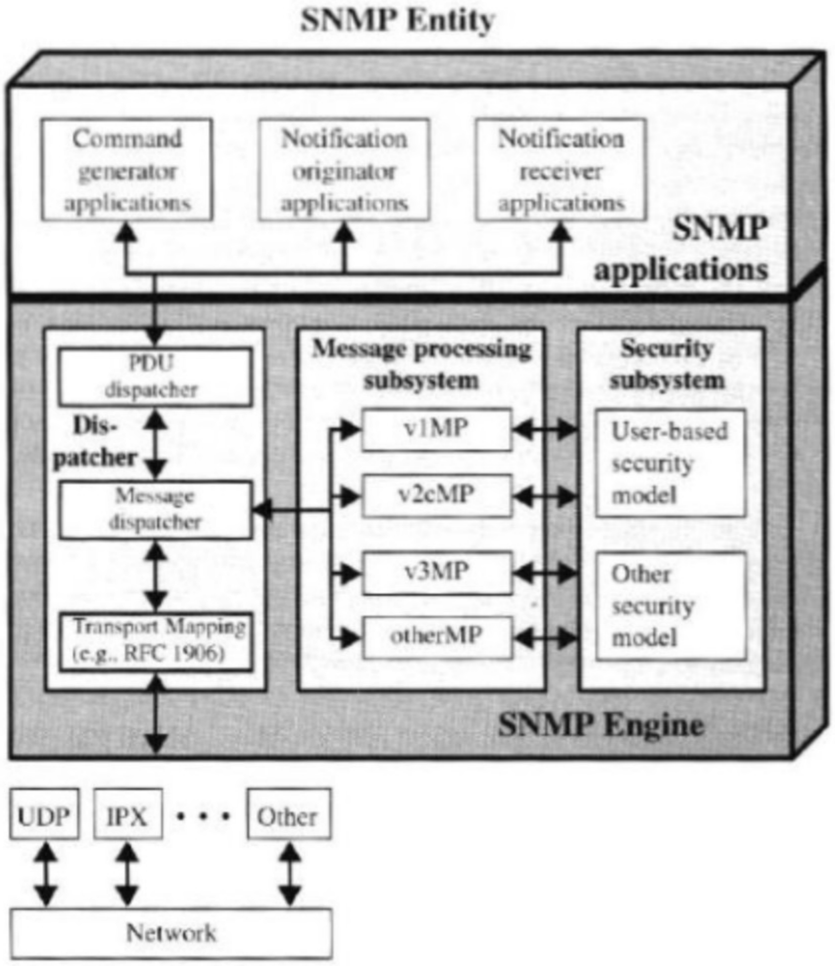
❖ **SNMP Entity**

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


SNMP Entity
Command generator applications
Notification originator applications
Notification receiver applications
SNMP applications
PDU dispatcher
Dis-patcher
Message dispatcher
Transport Mapping (e.g., RFC 1906)
Message processing subsystem
v1MP
v2cMP
v3MP
otherMP
Security subsystem
User-based security model
Other security model
SNMP Engine
UDP
IPX
Other
Network

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.

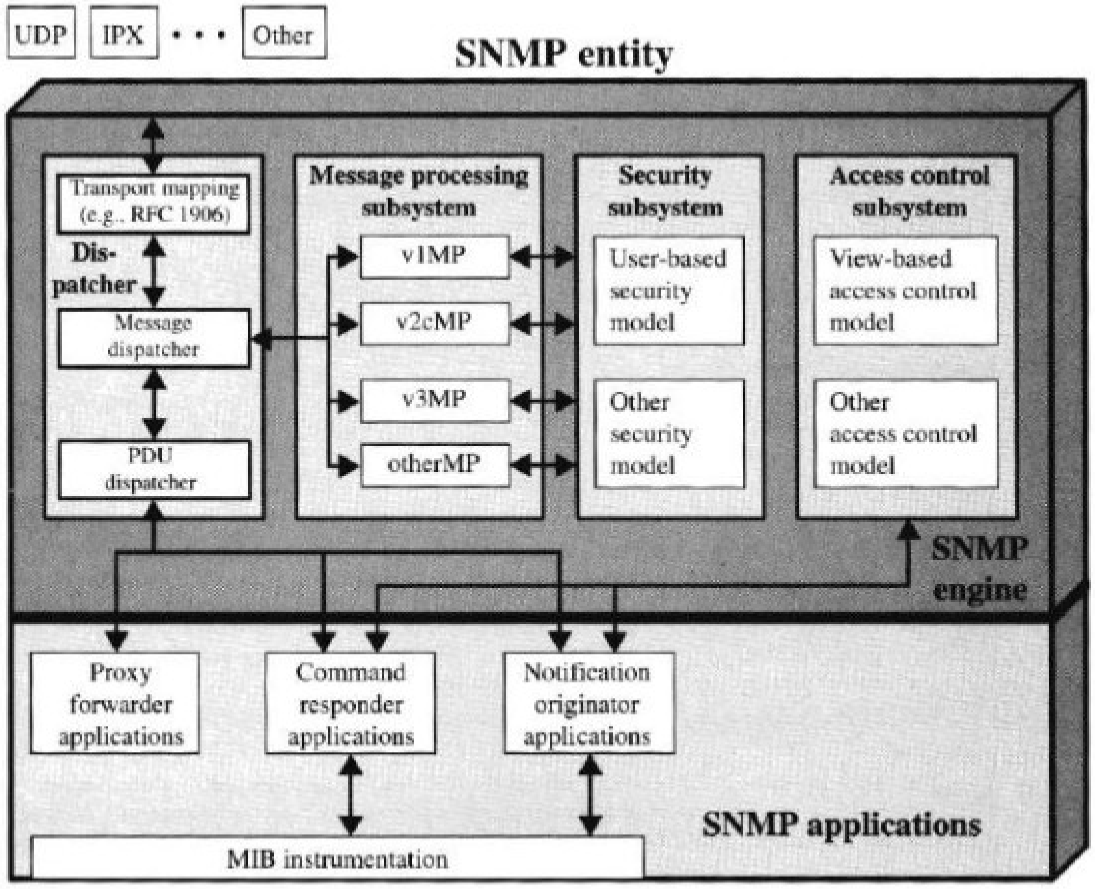
Notification Originator Application: In case of traditional SNMP manager, this application is responsible for starting the transmission of asynchronous messages like InformRequest PDU.

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


UDP
IPX
Other
SNMP entity
Transport mapping (e.g., RFC 1906)
Dis-patcher
Message dispatcher
PDU dispatcher
Message processing subsystem
v1MP
v2cMP
v3MP
otherMP
Security subsystem
User-based security model
Other security model
Access control subsystem
View-based access control model
Other access control model
SNMP engine
Proxy forwarder applications
Command responder applications
Notification originator applications
SNMP applications
MIB instrumentation

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.

UNIT - V

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

INTRUSION DETECTION:

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

· If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.

· An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

· Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the 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.

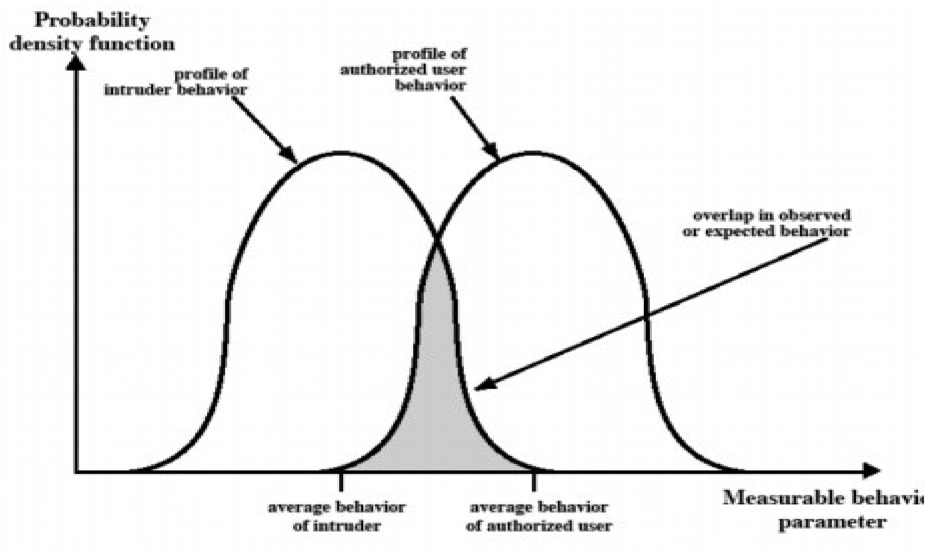

Probability density function
profile of intruder behavior
profile of authorized user behavior
overlap in observed or expected behavior
average behavior of intruder
average behavior of authorized user
Measurable behavi
parameter

Fig. 5.2.1Profilesofbehavior 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

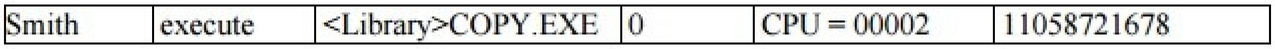
· **Resource-Usage**: A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read

o or written, processor time, I/O units used, session elapsed time).

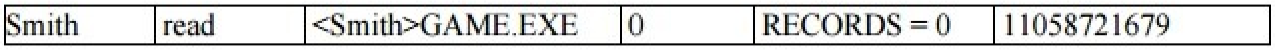
· **Time-Stamp**: Unique time-and-date stamp identifying when the action took place. Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command

COPY GAME.EXE TO <Library>GAME.EXE

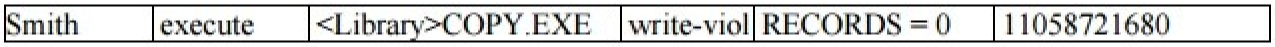
issued by Smith to copy an executable file GAME from the current directory to the <Library> directory. The following audit records may be generated:



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.

1.1 Statistical Anomaly Detection:

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

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

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

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on 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.

1.3 Rule-Based Intrusion Detection

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

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are 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:

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

2 The Base-Rate Fallacy

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

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating.

3 Distributed Intrusion Detection

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

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

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

Either a centralized or decentralized architecture can be used.

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

· **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.

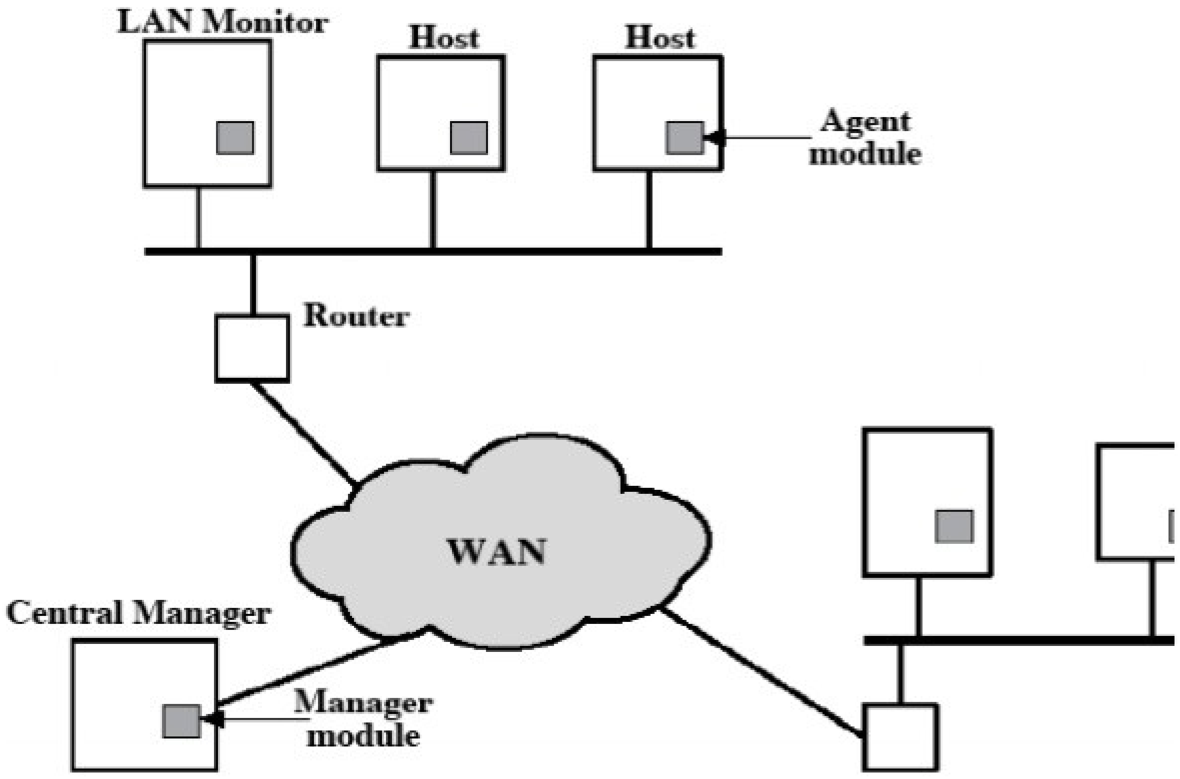


Fig. 5.2.3.1 Architecture of Distributed intrusion detection

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 lowestlevel, the agent scans for notable events that are of interest independent of any past events.
- Examplesinclude 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 atterns (signatures).

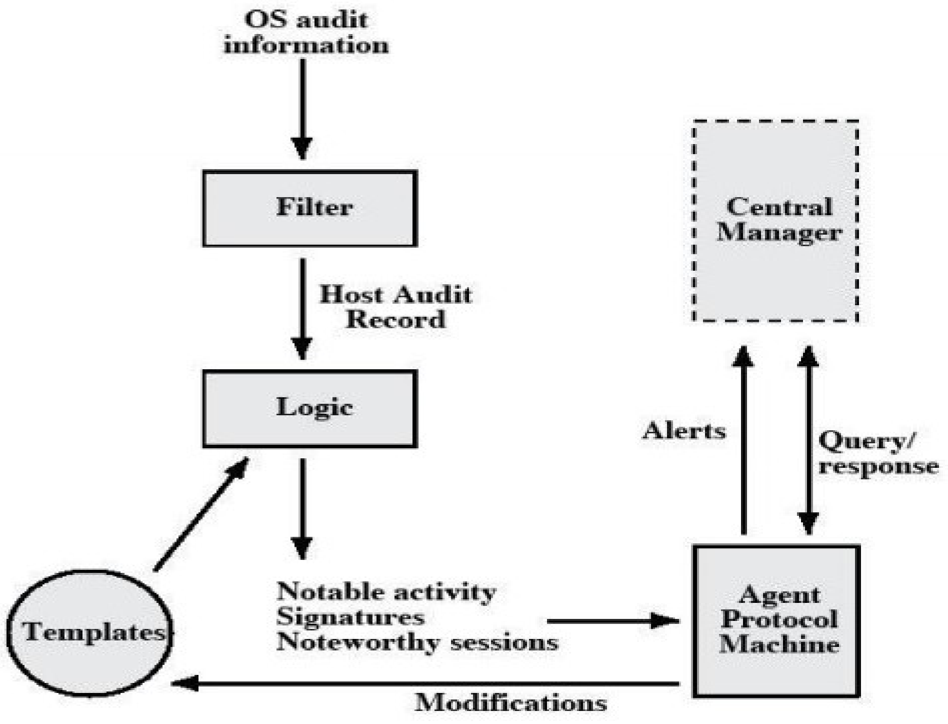
- Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.
- When suspicious activity is detected, an alert is sent to the central manager.
- The central manager includes 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.

4 Honeypots

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

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



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

5 Intrusion Detection Exchange Format

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

The outputs of this working group include the following:

a. A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements.

b. A common intrusion language specification, which describes data formats that satisfy the requirements.

c. A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised data formats relate to them.

PASSWORD MANAGEMENT

1. Password Protection

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

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

2. The Vulnerability of Passwords

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

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

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

The salt serves three purposes:

- 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.

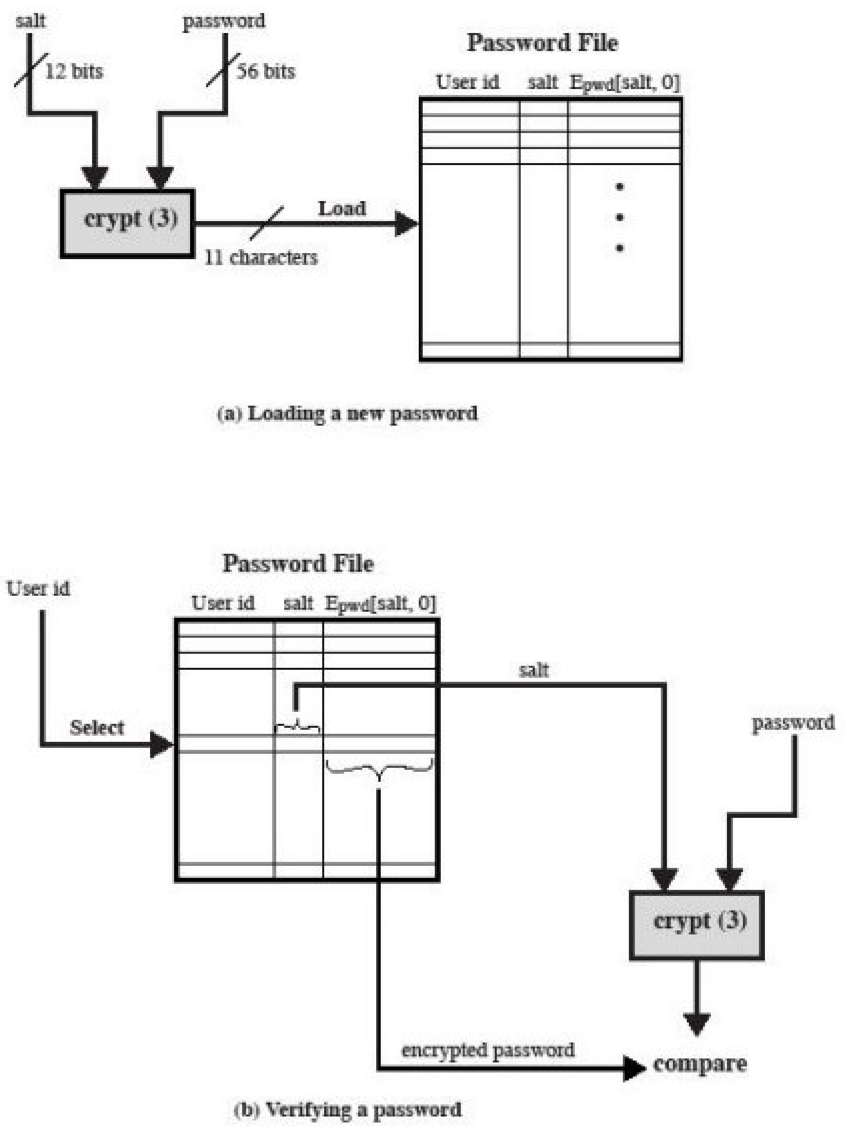


Fig. 5.3.2.1 UNIX Password Scheme

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.

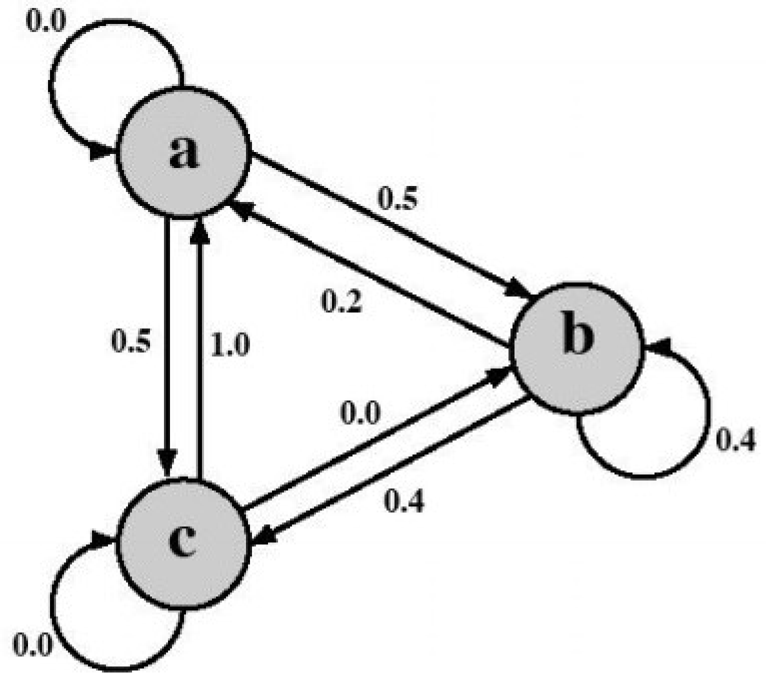
There are two problems with this approach:

- 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 kth-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: abbcacaba

e.g., string probably not from this language: aacccbaaa

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

1. Determine the frequency matrix f, where f(i, j, k) is the number of occurrences of the trigram consisting of the ith, jth, and kth character. For example, the password parsnips yields the trigrams par, ars, rsn, sni, nip, and ips.

2. For each bigram ij, calculate f(i, j,∞) as the total number of trigrams beginning with ij. For example, f(a, b,∞) would be the total number of trigrams of the form aba, abb, abc, and so on.

3. Compute the entries of T as follows:

T(i,j,k) = f(i, j, k) / f(i, j,∞)

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

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

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

where
X_j = jth word in password dictionary
D = number of words in password dictionary

The following procedure is then applied to the dictionary:

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

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

remains at 1.

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

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

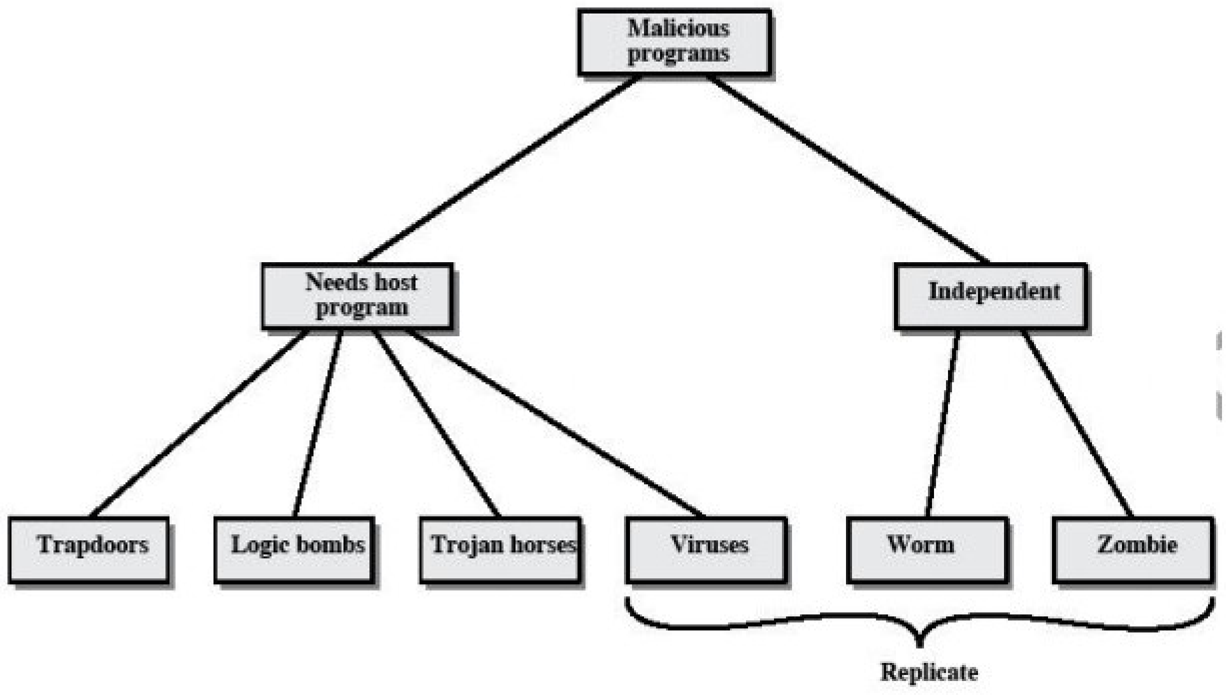


Figure 19.1 Taxonomy of Malicious Programs

Malicious software can be divided into two categories:

those that need a host program, and those that are independent.

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

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

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.

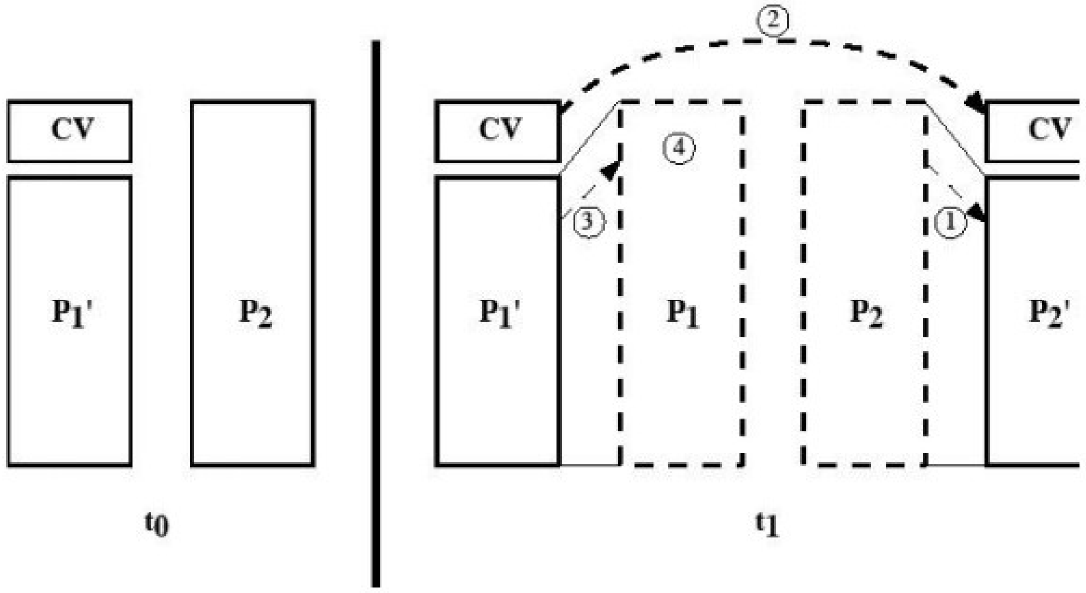
When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.

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.

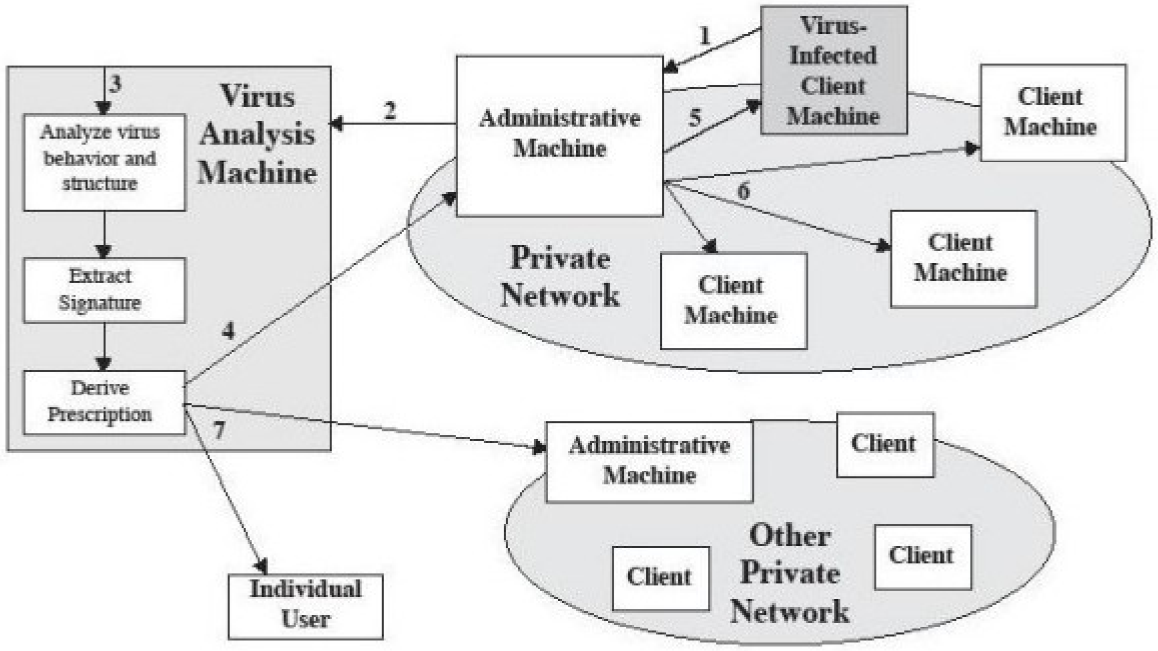
· **Emulation control module:** Controls the execution of the target code.

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:

1. Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.

2. Direction control – determines the direction in which particular service request may be initiated and allowed to flow through the firewall.

3. User control – controls access to a service according to which user is attempting to access it.

4. 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.

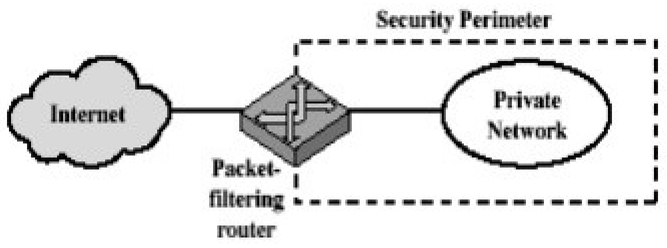
Types of firewalls

There are 3 common types of firewalls.

- Packet filters
- Application-level gateways
- 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:

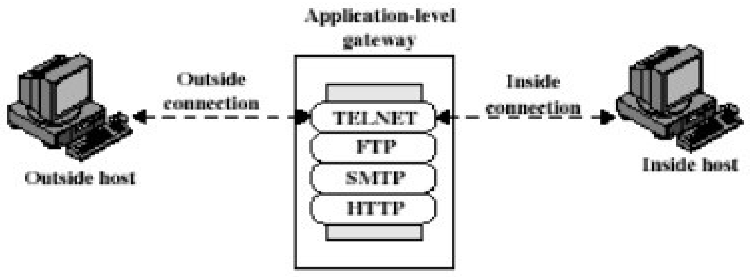


(a) Packet-filtering router

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.

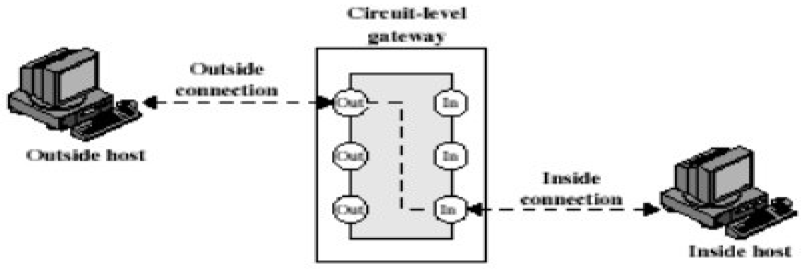
Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.



(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 :

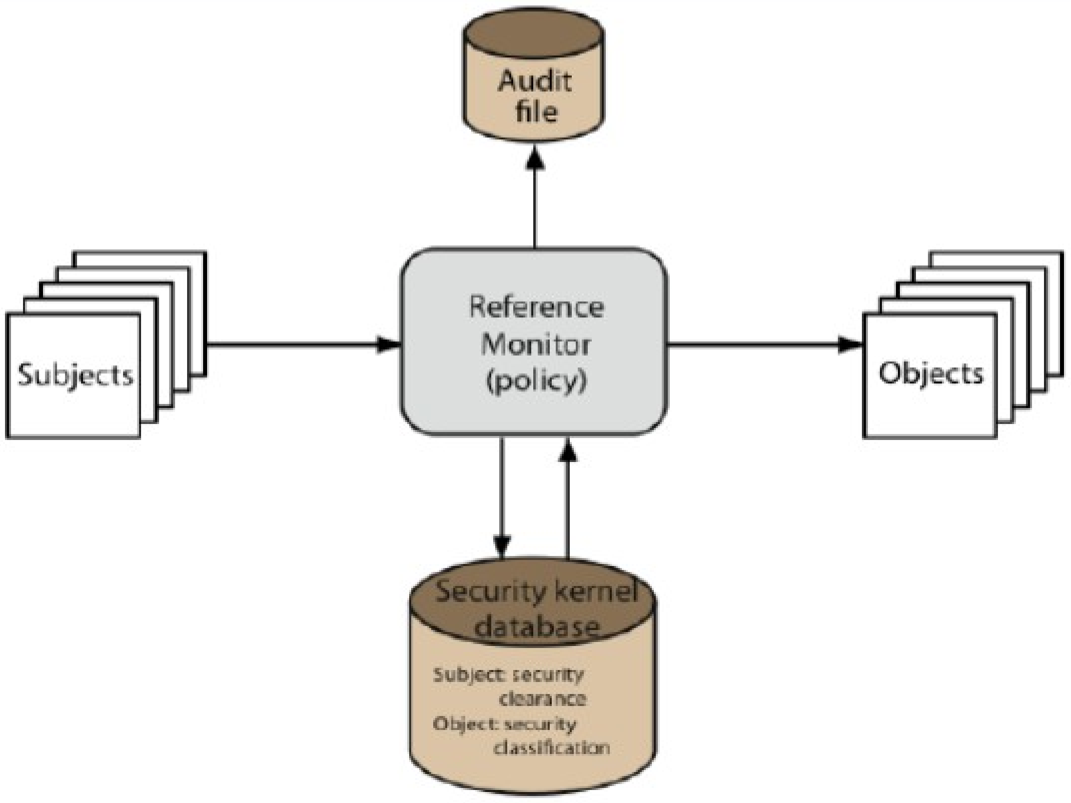
A widely applicable approach for protection of data and resources is based on levels of security. This is commonly found in military, where information is categorized as unclassified (U), confidential (C), secret (S), top secret (TS), or beyond. This concept is equally applicable in other areas, where 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**.


Audit file
Subjects
Reference Monitor (policy)
Objects
Security kernel database
Subject: security clearance
Object: security classification

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.

Common Criteria for Information Technology Security Evaluation

1 The CC permits comparability between the results of independent security evaluations. The CC does so by providing a common set of requirements for the security functionality of IT products and for assurance measures applied to these IT products during a security evaluation. These IT products may be implemented in hardware, firmware or software.

2 The evaluation process establishes a level of confidence that the security functionality of these IT products and the assurance measures applied to these IT products meet these requirements. The evaluation results may help consumers to determine whether these IT products fulfil their security needs.

3 The CC is useful as a guide for the development, evaluation and/or procurement of IT products with security functionality.

4 The CC is intentionally flexible, enabling a range of evaluation methods to be applied to a range of security properties of a range of IT products. Therefore users of the standard are cautioned to exercise care that this flexibility is not misused. For example, using the CC in conjunction with unsuitable evaluation methods, irrelevant security properties, or inappropriate IT products, may result in meaningless evaluation results.

5 Consequently, the fact that an IT product has been evaluated has meaning only in the context of the security properties that were evaluated and the evaluation methods that were used. Evaluation authorities are advised to carefully check the products, properties and methods to determine that an evaluation will provide meaningful results. Additionally, purchasers of evaluated products are advised to carefully consider this context to determine whether the evaluated product is useful and applicable to their specific situation and needs.

6 The CC addresses protection of assets from unauthorised disclosure, modification, or loss of use. The categories of protection relating to these three types of failure of security are commonly called confidentiality, integrity, and availability, respectively. The CC may also be applicable to aspects of IT security outside of these three. The CC is applicable to risks arising from human activities (malicious or otherwise) and to risks arising from non-human activities. Apart from IT security, the CC may be applied in other areas of IT, but makes no claim of applicability in these areas.

7 Certain topics, because they involve specialised techniques or because they are somewhat peripheral to IT security, are considered to be outside the scope of the CC. Some of these are identified below.

a) The CC does not contain security evaluation criteria pertaining to administrative security measures not related directly to the IT security functionality. However, it is recognised that significant security can often be achieved through or supported by administrative measures such as organisational, personnel, physical, and procedural controls. Introduction Page 12 of 106 Version 3.1 April 2017

b) The evaluation of some technical physical aspects of IT security such as electromagnetic emanation control is not specifically covered, although many of the concepts addressed will be applicable to that area.

c) The CC does not address the evaluation methodology under which the criteria should be applied. This methodology is given in the CEM.

d) The CC does not address the administrative and legal framework under which the criteria may be applied by evaluation authorities. However, it is expected that the CC will be used for evaluation purposes in the context of such a framework.

e) The procedures for use of evaluation results in accreditation are outside the scope of the CC. Accreditation is the administrative process whereby authority is granted for the operation of an IT product (or collection thereof) in its full operational environment including all of its non-IT parts. The results of the evaluation process are an input to the accreditation process. However, as other techniques are more appropriate for the assessments of non-IT related properties and their relationship to the IT security parts, accreditors should make separate provisions for those aspects.

f) The subject of criteria for the assessment of the inherent qualities of cryptographic algorithms is not covered in the CC. Should independent assessment of mathematical properties of cryptography be required, the evaluation scheme under which the CC is applied must make provision for such assessments.

8 ISO terminology, such as "can", "informative", "may", "normative", "shall" and "should" used throughout the document are defined in the ISO/IEC Directives, Part 2. Note that the term "should" has an additional meaning applicable when using this standard. See the note below. The following definition is given for the use of “should” in the CC. within normative text, “should” indicates “that among several.

9 should possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required.” (ISO/IEC Directives, Part 2). The CC interprets “not necessarily required” to mean that the choice of another possibility requires a justification of why the preferred option was not chosen.