DIGITAL NOTES

ON

INFORMATION SECURITY (R20A6251)

B. TECH III YEAR – II SEM (2023-2024)

PREPARED BY
T.SRINIDHI

MALLA REDDY COLLEGE OF ENGINEERING&TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956
(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

B.TECH - III- YEAR II-SEM-IT **L/T/P/C**

OPEN ELECTIVE - III **3/-/-/3**

(R20A6251) INFORMATION SECURITY

COURSE OBJECTIVES:

1. To understand the basic categories of threats to computers and networks.
2. To understand various cryptographic algorithms.
3. To apply authentication functions for providing effective security.
4. To analyze the application protocols to provide web security.
5. To understand the importance and implementation of Web security and Firewalls

UNIT- I

Attacks on Computers and Computer Security: Introduction, The need for security, Security approaches, Principles of security, Types of Security attacks, Security services, Security Mechanisms, A model for Network Security

Cryptography: Concepts and Techniques: Introduction, plain text and cipher text, substitution techniques, transposition techniques, encryption and decryption, symmetric and asymmetric key cryptography, steganography, key range and key size, possible types of attacks.

UNIT- II

Symmetric key Ciphers: Block Cipher principles & Algorithms(DES, AES), Block cipher modes of operation, Stream ciphers, RC4,Location and placement of encryption function, Key distribution **Asymmetric key Ciphers:** Principles of public key cryptosystems, Algorithms(RSA, Diffie-Hellman), Key Distribution.

UNIT-III

Message Authentication Algorithms and Hash Functions: Authentication requirements, Functions, Message authentication codes, Hash Functions, Secure hash algorithm, Whirlpool, HMAC, Digital signatures, **Authentication Applications:** Kerberos, X.509 Authentication Service, Public ―Key Infrastructure, Biometric Authentication

UNIT- IV

E-Mail Security: Pretty Good Privacy, S/MIME **IP Security:** IP Security overview, IP Security architecture, Authentication Header, encapsulating security payload, combining security associations, key management

UNIT- V

Web Security: Web security considerations, Secure Socket Layer and Transport Layer Security, Secure electronic transaction **Intruders, Virus and Firewalls:** Intruders, Intrusion

Detection, password management, Virus and related threats, Countermeasures, Firewall design principles, Types of firewalls **Case Studies on Cryptography and security:** Secure **Inter-Branch** Payment Transactions, Cross site Scripting Vulnerability, Virtual Elections

TEXT BOOKS:

1. Cryptography and Network Security : William Stallings, Pearson Education,4" Edition
2. Cryptography and Network Security : Atul Kahate, Mc Graw Hill, 2" Edition

REFERENCE BOOKS:

1. Cryptography and Network Security: C K Shyamala, N Harini, Dr T R Padmanabhan, Wiley India, lst Edition.
2. Cryptography and Network Security : Forouzan Mukhopadhyay, Mc Graw Hill, 2"d Edition
3. Information Security, Principles and Practice: Mark Stamp, Wiley India.
4. Principles of Computer Sceurity: WM.Arthur Conklin, Greg White, TMH
5. Introduction to Network Security: Neal Krawetz, CENGAGE Learning
6. Network Security and Cryptography: Bernard Menezes, CENGAGE Learning

COURSE OUTCOMES:

1. Students will be able to evaluate various security attacks and will gain understanding on services and mechanisms.
2. Students will understand the applications and working of various symmetric and asymmetric algorithms
3. Students will be able to identify information system requirements for both of them such as client and server.
4. Students will be able to understand the importance of IP security and key management.
5. Students will understand other types of threats and also the importance of web security

INDEX

UNIT - I

Attacks on Computers and Computer Security: *Introduction, The need for security, Security approaches, Principles of security, Types of Security attacks, Security services, Security Mechanisms, A model for Network Security*

Cryptography: Concepts and Techniques: *Introduction, plain text and cipher text, substitution techniques, transposition techniques, encryption and decryption, symmetric and asymmetric key cryptography, steganography, key range and key size, possible types of attacks.*

Introduction:

This is the age of universal electronic connectivity, where the activities like hacking, viruses, electronic fraud are very common. Unless security measures are taken, a network conversation or a distributed application can be compromised easily. Some simple examples are:

1. Online purchases using a credit/debit card.
2. A customer unknowingly being directed to a false website.
3. A hacker sending a message to a person pretending to be someone else.

Network Security has been affected by two major developments over the last several decades. First one is introduction of computers into organizations and the second one being introduction of distributed systems and the use of networks and communication facilities for carrying data between users & computers.

These two security deals with collection of tools designed to protect data and to thwart hackers. Network security measures are needed to protect data during transmission. But keep in mind that, it is the information and our ability to access that information that we are really trying to protect and not the computers and networks.

The Need for Security

Basic Concepts

Most initial computer applications had *no* or at best, *very little* security. This continued for a number of Years until the importance of data was truly realized. Until then, computer data was considered to be useful, but not something to be protected. When computer applications were developed to handle Financial and personal data, the real need for security was felt like never before. People realized that data On computers was an extremely important aspect of modern life. Therefore, various areas in security Began to gain prominence. Two typical examples of such security mechanisms were as follows:

- Provide a user id and password to every user and use that information to authenticate a user
- Encode information stored in the databases in some fashion so that it is not visible to users who do not have the right permissions

Organizations employed their own mechanisms in order to provide for these kinds of basic security Mechanisms. As technology improved, the communication infrastructure became extremely mature and, Newer and newer applications began to be developed for various user demands and needs. Soon, people Realized that the basic security measures were not quite enough.

Furthermore, the Internet took the world by storm and there were many examples of what could Happen if there was insufficient security built in applications developed for the Internet. Figure 1.1 shows such an example of what can happen when you use your credit card for making purchases over the Internet. From the user's computer, the user details such as user id, order details such as order id and item id, and payment details such as credit card information travel across the Internet to the server (i.e. to the merchant's computer). The merchant's server stores these details in its database.

There are various security holes here. First of all, an intruder can capture the credit card details as they travel from the client to the server. If we somehow protect this transit from an intruder's attack, it still does not solve our problem. Once the merchant receives the credit card details and validates them so as to process the order and later obtain payments, the merchant stores the credit card details into its database.

Now, an attacker can simply succeed in accessing this database and gain access to all the credit card numbers stored therein! One Russian attacker (called as Maxim) actually managed to intrude into a merchant

Internet site and obtained 300,000 credit card numbers from its database. He then attempted extortion by demanding protection money ($100,000) from the merchant.

The merchant refused to oblige. Following this, the attacker published about 25,000 of the credit card numbers on the Internet! Some banks reissued all the credit cards at a cost of $20 per card and others forewarned their customers about unusual entries in their statements.

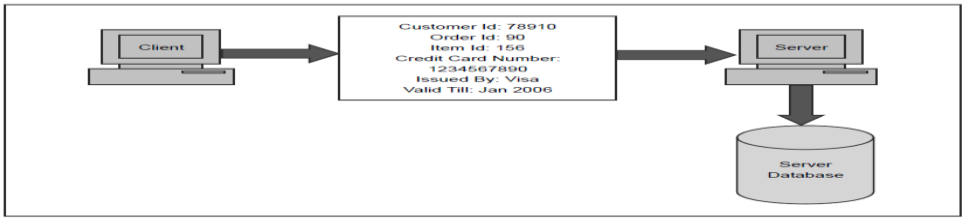


Fig. 1.1 *Example of information traveling from a client to a server over the Internet*

Such attacks could obviously lead to great losses – both in terms of finance and goodwill. Generally, it takes $20 to replace a credit card. Therefore, if a bank has to replace 3,00,000 such cards, the total cost of such an attack is about $6 million! How nice it would have been, if the merchant in the example just discussed had employed proper security measures!

Of course, this was just one example. Several such cases have been reported in the last few months And the need for proper security is being felt increasingly with every such attack. In another example of this, in 1999, a Swedish hacker broke into Microsoft's Hotmail Web site and created a mirror site. This site allowed anyone to enter any Hotmail user's email id and read her emails!

In 2005 as independent survey was conducted to invite people's opinions about the losses that occur due to successful attacks on security. The survey pegged the losses at an average of $455,848,000. Next year, this figure reduced to $201,757,340!

Modern Nature of Attacks

If we attempt to demystify technology, we would realize that computer-based systems are not all that different from what happens in the real world. Differences in computer-based systems are mainly due to the speed at which things happen and the accuracy that we get, as compared to the traditional world.

We can highlight a few salient features of the modern nature of attacks, as follows:

• ***Automating attacks*** The speed of computers make several attacks worthwhile. For example, in the real world, suppose that someone manages to create a machine that can produce counterfeit coins. Would that not bother authorities? It certainly would. However, producing so many coins on a mass scale may not be that much economical compared to the return on that investment! How many such coins would the attacker be able to get into the market so rapidly? This is quite different with computers. They are quite efficient and happy in doing routine, mundane and repetitive tasks. For example, they would excel in somehow stealing a very low amount (say half a dollar or Rupees 20) from a million bank accounts in a matter of a few minutes. This would give the attacker a half million dollars possibly without any major

Complaints! This is shown in Fig. 1.2.

The morale of the story is:

"Humans dislike mundane and repetitive tasks. Automating them can cause destruction or nuisance quite rapidly."

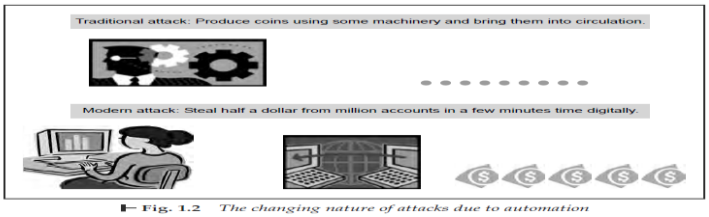


Fig. 1.2 *The changing nature of attacks due to automation*

Privacy concerns: Collecting information about people and later (mis)using it is turning out to be a huge problem, these days. The so called *data mining* applications gather, process and tabulate all sorts of details about individuals. People can then illegally sell this information. For example, companies like Experian (formerly TRW), Transunion and Equifax maintain credit history of individuals in the USA.

Similar trends are seen in the rest of the world. These companies have volumes of information about a majority of citizens of that country. These companies can collect, collate, polish and format all sorts of information to whosoever is ready to pay for that data! Examples of information that can come out of this are: which store the person buys more from, which restaurant she eats in, where she goes for vacations frequently and

so on! Every company (e.g. shopkeepers, banks, airlines, insurers) are collecting and processing a mind-boggling amount of information about us, without we realizing when and how it is going to be used.

Distance does not matter: Thieves would earlier attack banks, because banks had money. Banks do not have money today! Money is in digital form inside computers and moves around by using computer networks. Therefore, a modern thief would perhaps not like to wear a mask and attempt a robbery! Instead, it is far easier and cheaper to attempt an attack on the computer systems of the bank, sitting at home! It may be far prudent for the attacker to break into the bank's servers or steal credit card/ATM information from the comforts of her home or place of work. This is illustrated in Fig. 1.3.

In 1995, a Russian hacker broke into Citibank's computers remotely, stealing $ 12 million. Although the attacker was traced, it was very difficult to get him extradited for the court case.

Security Approaches: Trusted Systems

A **trusted system** is a computer system that can be trusted to a specified extent to enforce a specified security policy. Trusted systems were initially of primary interest to the military. However, these days, the concept has spanned across various areas, most prominently in the banking and financial community, but the concept never caught on. Trusted systems often use the term **reference monitor**. This is an entity that is at the logical heart of the computer system. It is mainly responsible for all the decisions related to access controls. Naturally, following are the expectations from the reference monitor:

(a) It should be tamperproof

(b) It should always be invoked

(c) It should be small enough so that it can be independently tested

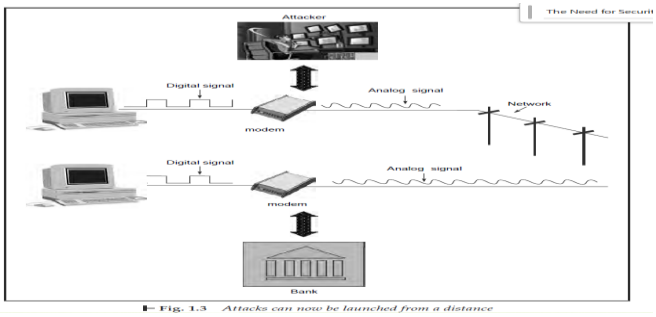


Fig. 1.3 *Attacks can now be launched from a distance*

In their 1983 *Orange Book* (also called as the *Trusted Computer System Evaluation Criteria (TCSEC)*), the National Security Agency (NSA) of the US Government defined a set of *evaluation classes*. These described the features and assurances that the user could expect from a trusted system.

The highest levels of assurance were provided by significant efforts directed towards reduction of the size of the trusted computing base or TCB. In this context, TCB was defined as a combination of hardware, software and firmware responsible for enforcing the system's security policy. Minimum the TCB, higher is assurance. However, this raises an inherent problem (quite similar to the decisions related to the designing of operating systems). If we make the TCB as small as possible, the surrounding hardware, software and firmware is likely to be quite big!

The mathematical foundation for trusted systems was provided by two relatively independent yet interrelated works. In the year 1974, David Bell and Leonard LaPadula of MITRE devised a technique called as the **Bell-LaPadula model.** In this model, a highly trustworthy computer system is designed as a collection of objects and subjects. Objects are passive repositories or destinations for data, such as files, disks, printers, etc. Subjects are active entities, such as users, processes or threads operating on behalf of those users. Subjects cause information to flow among objects.

Around the same time, Dorothy Denning at Purdue University was preparing for her doctorate. It dealt with *lattice-based information flows* in computer systems. A mathematical *lattice* is a partially ordered set, in which the relationship between any two vertices is either *dominates*, *is dominated by* or *neither*. She devised a generalized notion of *labels* — similar to the full security markings on classified military documents. Examples of this are TOP SECRET.

Later, Bell and LaPadula integrated Denning's theory into their MITRE technical report, which was titled *Secure Computer System: Unified Exposition and Multics Interpretation*. Here, *labels* attached to *objects* represented the sensitivity of data contained within the *object*. Interestingly, the Bell-LaPadula model talks only about *confidentiality* or *secrecy* of information. It does not talk about the problem of *integrity* of information.

Security Models

An organization can take several approaches to implement its security model. Let us summarize these approaches.

- ***No security*** In this simplest case, the approach could be a decision to implement no security at all.
- ***Security through obscurity*** In this model, a system is secure simply because nobody knows about its existence and contents. This approach cannot work for too long, as there are many ways an attacker can come to know about it.
- ***Host security*** In this scheme, the security for each host is enforced individually. This is a very safe approach, but the trouble is that it cannot scale well. The complexity and diversity of modern sites/ organizations makes the task even harder.
- ***Network security*** Host security is tough to achieve as organizations grow and become more diverse. In this technique, the focus is to control network access to various hosts and their services, rather than individual host security. This is a very efficient and scalable model.

Security Management Practices

Good **security management practices** always talk of a **security policy** being in place. Putting a security policy in place is actually quite tough. A good security policy and its proper implementation go a long way in ensuring adequate security management practices. A good security policy generally takes care of four key aspects, as follows:

- ***Affordability*** Cost and effort in security implementation.
- ***Functionality*** Mechanism of providing security.
- ***Cultural issues*** Whether the policy gels well with people's expectations, working style and beliefs.
- ***Legality*** Whether the policy meets the legal requirements.

Once a security policy is in place, the following points should be ensured.

a) Explanation of the policy to all concerned.
b) Outline everybody's responsibilities.
c) Use simple language in all communications.
d) Establishment of accountability.
e) Provision for exceptions and periodic reviews.

Principles of Security

Having discussed some of the attacks that have occurred in real life, let us now classify the principles related to security. This will help us understand the attacks better and also help us in thinking about the possible solutions to tackle them. We shall take an example to understand these concepts. Let us assume that a person A wants to send a check worth $100 to another person B. Normally, what are the factors that A and B will think of, in such a case? A will write the check for $100, put it inside an envelope and send it to B.

- ✓ A will like to ensure that no one except B gets the envelope and even if someone else gets it, she does not come to know about the details of the check. This is the principle of **confidentiality**.
- ✓ A and B will further like to make sure that no one can tamper with the contents of the check (such as its amount, date, signature, name of the payee, etc.). This is the principle of **integrity**.
- ✓ B would like to be assured that the check has indeed come from A and not from someone else posing as A (as it could be a fake check in that case). This is the principle of **authentication**.
- ✓ What will happen tomorrow if B deposits the check in her account, the money is transferred from A's account to B's account and then A refuses having written/sent the check? The court of law will use A's signature to disallow A to refute this claim and settle the dispute. This is the principle of **non-repudiation**.

These are the four chief principles of security. There are two more, **access control** and **availability**, which are not related to a particular message, but are linked to the overall system as a whole. We shall discuss all these security principles in the next few sections.

Confidentiality

The principle of *confidentiality* specifies that only the sender and the intended recipient(s) should be able to access the contents of a message. Confidentiality gets compromised if an unauthorized person is able to access a message. Example of compromising the confidentiality of a message is shown in Fig. 1.4. Here, the user of computer A sends a message to user of computer B. (Actually, from here onwards, we

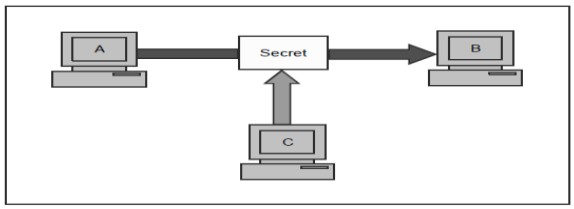


Fig. 1.4 *Loss of confidentiality*

shall use the term A to mean the *user* A, B to mean *user* B, etc. although we shall just show the computers of user A, B, etc.). Another user C gets access to this message, which is not desired and therefore, defeats the purpose of confidentiality. Example of this could be a confidential email message sent by A to B, which is accessed by C without the permission or knowledge of A and B. This type of attack is called as **interception**.

Interception causes loss of message confidentiality.

Authentication

Authentication mechanisms help establish proof of identities. The authentication process ensures that the origin of a electronic message or document is correctly identified. For instance, suppose that user C sends an electronic document over the Internet to user B. However, the trouble is that user C had posed as user A when she sent this document to user B. How would user B know that the message has come from user C, who is posing as user A? A real life example of this could be the case of a user C, posing as user A, sending a funds transfer request (from A's account to C's account) to bank B. The bank might happily transfer the funds from A's account to C's account – after all, it would think that user A has requested for the funds transfer! This concept is shown in Fig. 1.5. This type of attack is called as **fabrication**.

Fabrication is possible in absence of proper authentication mechanisms.

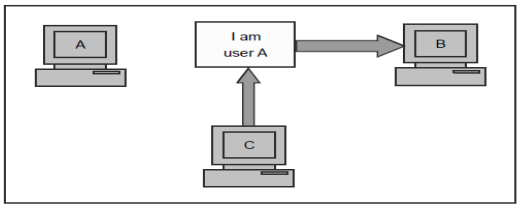


Fig. 1.5 *Absence of authentication*

Integrity

When the contents of a message are changed after the sender sends it, but before it reaches the intended recipient, we say that the *integrity* of the message is lost. For example, suppose you write a check for $100 to pay for the goods bought from the US. However, when you see your next account statement, you are startled to see that the check resulted in a payment of $1000! This is the case for loss of message integrity. Conceptually, this is shown in Fig. 1.6. Here, user C tampers with a message originally sent by user A, which is actually destined for user B. User C somehow manages to access it, change its contents and send the changed message to user B. User B has no way of knowing that the contents of the message were changed after user A had sent it. User A also does not know about this change. This type of attack is called as **modification**.

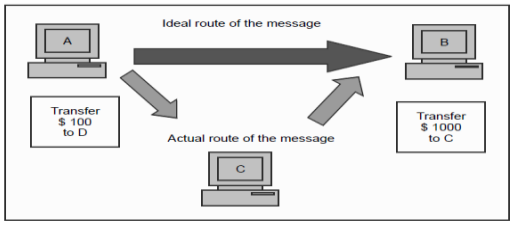


Fig. 1.6 *Loss of integrity*

Modification causes loss of message integrity.

Non-repudiation

There are situations where a user sends a message and later on refuses that she had sent that message. For instance, user A could send a funds transfer request to bank B over the Internet. After the bank performs the funds transfer as per A's instructions, A could claim that she never sent the funds transfer instruction to the bank! Thus, A repudiates or denies, her funds transfer instruction. The principle of *nonrepudiation* defeats such possibilities of denying something, having done it. This is shown in Fig. 1.7.

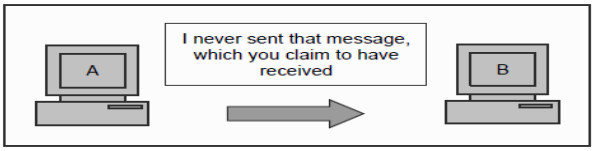


Fig. 1.7 *Establishing non-repudiation*

Access Control

The principle of *access control* determines *who* should be able to access *what*. For instance, we should be able to specify that user A can view the records in a database, but cannot update them. However, user B might be allowed to make updates as well. An access control mechanism can be set up to ensure this. Access control is broadly related to two areas: *role management* and *rule management*. Role management concentrates on the user side (which user can do what), whereas rule management focuses on the resources side (which resource is accessible and under what circumstances). Based on the decisions taken here, an access control matrix is prepared, which lists the users against a list of items they can access (e.g. it can say that user A can write to file X, but can only update files Y and Z). An **Access Control List (ACL)** is a subset of an access control matrix.

Access control specifies and controls who can access what.

Availability

The principle of *availability* states that resources (i.e. information) should be available to authorized parties at all times. For example, due to the intentional actions of an unauthorized user C, an authorized user A may not be able to contact a server computer B, as shown in Fig. 1.8. This would defeat the principle of availability. Such an attack is called as **interruption**.

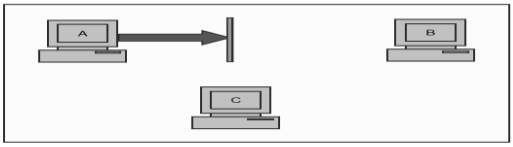


Fig. 1.8 *Attack on availability*

Interruption puts the availability of resources in danger.

1.4.7 Ethical and Legal Issues

Many ethical and legal issues in computer security systems seem to be in the area of the individual's right to privacy versus the greater good of a larger entity (e.g. a company, society, etc.) For example, tracking how employees use computers, crowd surveillance, managing customer profiles, tracking a person's travel with a passport, location tracking so as to spam cell phone with text message advertisements and so on. A key concept in resolving this issue is to find out is a person's expectation of privacy.

Classically, the ethical issues in security systems are classified into the following four categories:

- ✓ Privacy – This deals with the right of an individual to control personal information.
- ✓ Accuracy – This talks about the responsibility for the authenticity, fidelity and accuracy of information.
- ✓ Property – Here we find out the owner of the information. We also talk about who controls access.
- ✓ Accessibility – This deals with the issue of the type of information an organization has the right to collect. And in that situation, it also expects to know the measures which will safeguard against any unforeseen eventualities.

SECURITY ATTACK: *Any action that compromises the security of information owned by an organization.*

- Information security is about how to prevent attacks, or failing that, to detect attacks on information-based systems
- often *threat* & *attack* used to mean same thing have a wide range of attacks

Types of attacks

- Passive Attack
- Active Attack

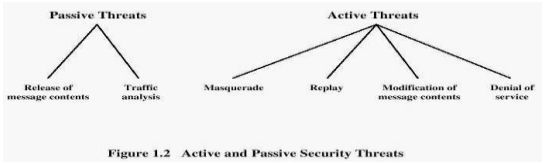
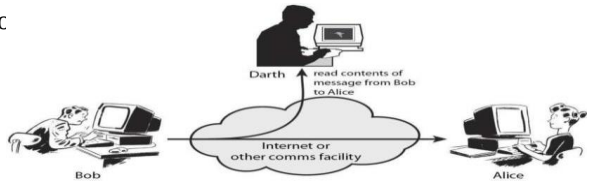


Figure 1.2 Active and Passive Security Threats

Passive Attack:

A passive attack is a network attack in which a system is monitored and sometimes scanned for open ports and vulnerabilities. The purpose of a passive attack is to gain information about the system being targeted; it does not

invc



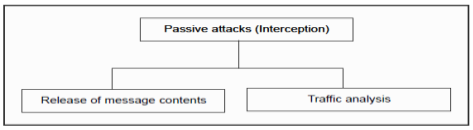
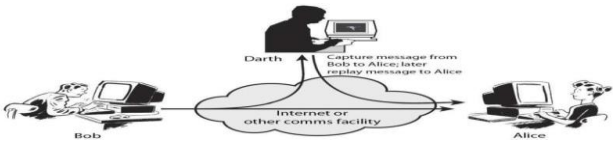


Fig. 1.11 *Passive attacks*

Active attack:

An active attack is a network exploit in which a hacker attempts to make changes to data on the target or data en route to the target.

- ✓ There are several different types of active attacks. However, in all cases, the threat actor takes some sort of action on the data in the system or the devices the data resides on.
- ✓ Attackers may attempt to insert data into the system or change or control data that is already in the system



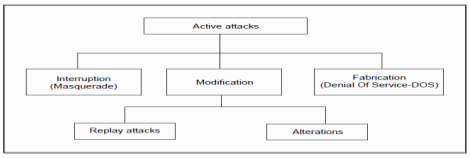


Fig. 1.12 *Active attacks*

INTERRUPTION

An asset of the system is destroyed or becomes unavailable or unusable. It is an attack on availability.

- ✓ Examples: Destruction of some hardware
  - Jamming wireless signals
  - Disabling file management systems

INTERCEPTION

An unauthorized party gains access to an asset. Attack on confidentiality.

Examples:

Wiretapping to capture data in a network. Illicitly copying data or programs Eavesdropping

MODIFICATION

When an unauthorized party gains access and tampers an asset. Attack is on Integrity.

Examples:

Changing data file

Altering a program and the contents of a message

FABRICATION

An unauthorized party inserts a counterfeit object into the system. Attack on Authenticity. Also called impersonation

Examples:

Hackers gaining access to a personal email and sending message Insertion of records in data files

Insertion of spurious messages in a network

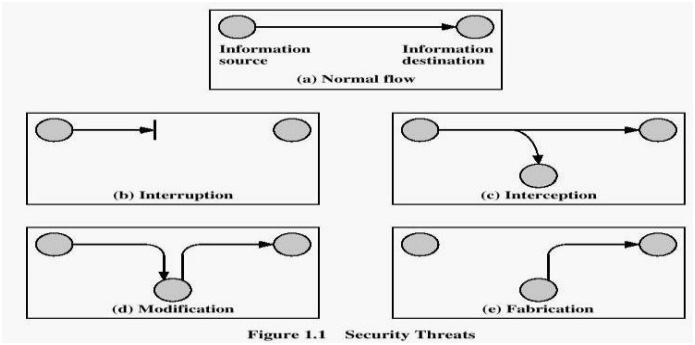


Figure 1.1 Security Threats

SECURITY SERVICES:

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

Confidentiality

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

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

Authentication

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

- ✓ **Peer entity authentication:** Verifies the identities of the peer entitiesinvolved in communication. Provides use at time of connection establishment and during data transmission. Provides confidence against a masquerade or a replay attack
- ✓ **Data origin authentication:** Assumes the authenticity of source of data unit, but does not provide protection against duplication or modification of data units. Supports applications like electronic mail, where no prior interactions take place between communicating entities.

Integrity

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

- **Connection-Oriented Integrity Service:** This service deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering or replays. Destruction of data is also covered here. Hence, it attends to both message stream modification and denial of service.
- **Connectionless-Oriented Integrity Service:** It deals with individual messages regardless of larger context, providing protection against message modification only.

An integrity service can be applied with or without recovery. Because it is related to active attacks, major concern will be detection rather than prevention.

If a violation is detected and the service reports it, either human intervention or automated recovery machines are required to recover.

Non-repudiation

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

Access Control

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

Availability

It is defined to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity. The availability can significantly be affected by a variety of attacks, some

amenable to automated counter measures i.e authentication and encryption and others need some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

SECURITY MECHANISMS

According to X.800, the security mechanisms are divided into those implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. X.800 also differentiates reversible & irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted, whereas irreversible encipherment include hash algorithms and message authentication codes used in digital signature and message authentication applications

Specific Security Mechanisms

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

- ✓ **Encipherment:** It refers to the process of applying mathematical algorithms for converting data into a form that is not intelligible. This depends on algorithm used and encryption keys.
- ✓ **Digital Signature:** The appended data or a cryptographic transformation applied to any data unit allowing to prove the source and integrity of the data unit and protect against forgery.
- ✓ **Access Control:** A variety of techniques used for enforcing access permissions to the system resources.
- ✓ **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- ✓ **Authentication Exchange:** A mechanism intended to ensure the identity of an entity by means of information exchange.
- ✓ **Traffic Padding:** The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
- ✓ **Routing Control:** Enables selection of particular physically secure routes for certain data and allows routing changes once a breach of security is suspected.
- ✓ **Notarization:** The use of a trusted third party to assure certain properties of a data exchange

Pervasive Security Mechanisms

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

- ✓ **Trusted Functionality:** That which is perceived to b correct with respect to some criteria **Security Level:** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
- ✓ **Event Detection:** It is the process of detecting all the events related to network security.
- ✓ **Security Audit Trail:** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
- ✓ **Security Recovery:** It deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

MODEL FOR NETWORK SECURITY

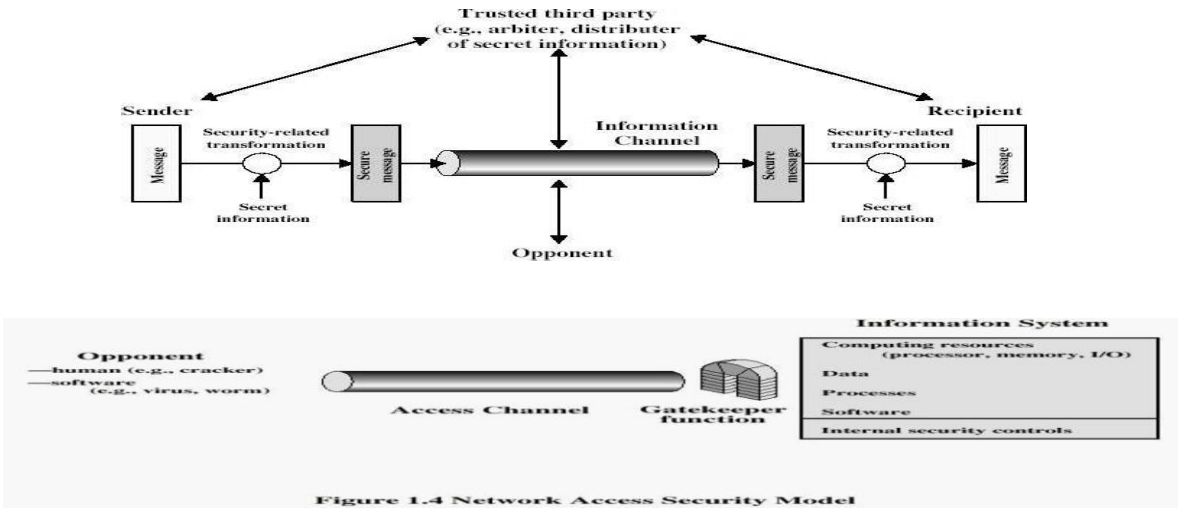


Figure 1.4 Network Access Security Model

Cryptography: Concepts and Techniques:

Data is transmitted over network between two communicating parties, who must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination by use of communication protocols by the two parties. Whenever an opponent presents a threat to confidentiality, authenticity of information, security aspects come into play. Two components are present in almost all the security providing techniques.

A security-related transformation on the information to be sent making it unreadable by the opponent, and the addition of a code based on the contents of the message, used to verify the identity of sender.

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

A trusted third party may be needed to achieve secure transmission. It is responsible for distributing the secret information to the two parties, while keeping it away from any opponent. It also may be needed to settle disputes between the two parties regarding authenticity of a message transmission. The 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 Various other threats to information system like unwanted access still exist.

The existence of hackers attempting to penetrate systems accessible over a network remains a concern. Another threat is placement of some logic in computer system affecting various applications and utility programs. This inserted code presents 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 inserted into the system by means of a disk or also across the network. The security mechanisms needed to cope with unwanted access fall into two broad categories.

Some basic terminologies used

- **CIPHER TEXT** - the coded message
- **CIPHER** - algorithm for transforming plaintext to cipher text
- **KEY** - info used in cipher known only to sender/receiver
- **ENCIPHER (ENCRYPT)** - converting plaintext to cipher text
- **DECIPHER (DECRYPT)** - recovering cipher text from plaintext
- **CRYPTOGRAPHY** - study of encryption principles/methods
- **CRYPTANALYSIS (CODEBREAKING)** - the study of principles/ methods of deciphering cipher text *without* knowing key**CRYPTOLOGY** - the field of both cryptography and cryptanalysis

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 abased 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.

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.

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

The scheme explained earlier (of replacing an alphabet with the one three places down the line) was first proposed by Julius Caesar and is termed as **Caesar Cipher**. It was the first example of substitution cipher. In the substitution cipher technique, the characters of a plain text message are replaced by other

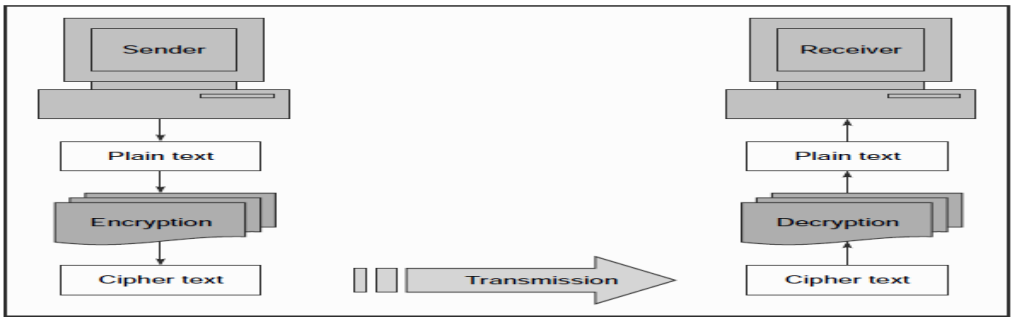


Fig. 2.7 *Elements of a cryptographic operation*

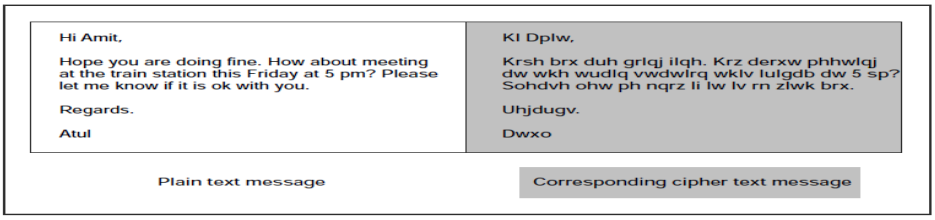


Fig. 2.8 *Example of a plain text message being transformed into cipher text*

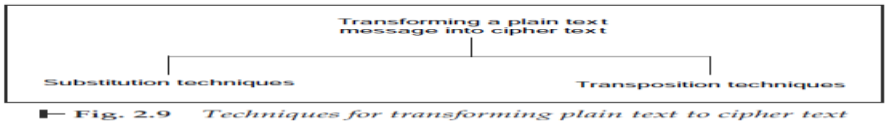


Fig. 2.9 *Techniques for transforming plain text to cipher text*

characters, numbers or symbols. Caesar Cipher is a special case of substitution techniques wherein each alphabet in a message is replaced by an alphabet three places down the line. For instance, using the Caesar Cipher, the plain text ATUL will become cipher text DWXO.

> *In the substitution cipher technique, the characters of a plain text message are replaced by other characters, numbers or symbols.*

Clearly, the Caesar Cipher is a very weak scheme of hiding plain text messages. All that is required to break the Caesar Cipher is to do the reverse of the Caesar Cipher process – i.e. replace each alphabet in a cipher text message produced by Caesar Cipher with the alphabet that is three places up the line. Thus, to work backwards, take a cipher text produced by Caesar Cipher and replace each A with X, B with Y, C with Z, D with A, E with B and so on. The simple algorithm required to break Caesar Cipher can be summarized as shown in Fig. 2.10.

1. Read each alphabet in the cipher text message, and search for it in the second row of the *replacement* table (i.e. the second row of the table).
2. When a match is found, replace that alphabet in the cipher text message with the corresponding alphabet in the same column but the first row of the table (e.g. if the alphabet in cipher text is J, replace it with G).
3. Repeat the process for all alphabets in the cipher text message.

Fig. 2.10 *Algorithm to break Caesar Cipher*

The process shown above will reveal the original plain text. Thus, given a cipher text message *L ORYH BRX*, it is easy to work backwards and obtain the plain text *I LOVE YOU* as shown in Fig. 2.11.

Cipher text	L		O	R	Y	H		B	R	X
Plain text	I		L	O	V	E		Y	O	U

Fig. 2.11 *Example of breaking Caesar Cipher*

Modified Version of Caesar Cipher:

Caesar Cipher is good in theory, but not so good in practice. Let us now try and complicate the Caesar Cipher to make an attacker's life difficult. How can we generalize Caesar Cipher a bit more? Let us assume that the cipher text alphabets corresponding to the original plain text alphabets may not necessarily be three places down the line, but instead, can be *any* places down the line. This can complicate matters a bit.

Thus, we are now saying that an alphabet A in plain text would not necessarily be replaced by D. It can be replaced by any valid alphabet, i.e. by E or by F or by G and so on. Once the replacement scheme is decided, it would be constant and will be used for all other alphabets in that message. As we know, the English language contains 26 alphabets. Thus, an alphabet A can be replaced by any *other* alphabet in the English alphabet set, (i.e. B through Z). Of course, it does not make sense to replace an alphabet by itself (i.e. replacing A with A). Thus, for each alphabet, we have 25 possibilities of replacement. Hence, to break a message in the modified version of Caesar Cipher, our earlier algorithm would not work. Let us write a new algorithm to break this version of Caesar Cipher, as shown in Fig. 2.12.

1. Let k be a number equal to 1.
2. Read the complete cipher text message.
3. Replace each alphabet in the cipher text message with an alphabet that is k positions down the order.
4. Increment k by 1.
5. If k is less than 26, then go to Step 2. Otherwise, stop the process.
6. The original text message corresponding to the cipher text message is one of the 25 possibilities produced by the above steps.

Fig. 2.12 *Algorithm to break the modified Caesar cipher*

2.3.3 Mono-alphabetic Cipher

The major weakness of the Caesar Cipher is its predictability. Once we decide to replace an alphabet in a plain text message with an alphabet that is *k* positions up or down the order, we replace all other alphabets in the plain text message with the same technique. Thus, the cryptanalyst has to try out a maximum of 25 possible attacks and she is assured of success.

Now imagine that rather than using a uniform scheme for all the alphabets in a given plain text message, we decide to use random substitution. This means that in a given plain text message, each A can be replaced by any other alphabet (B through Z), each B can also be replaced by any other random alphabet (A or C through Z) and so on. The crucial difference being, there is no relation between the replacement of B and replacement of A. That is, if we have decided to replace each A with D, we need not necessarily replace each B with E – we can replace each B with any other character!

Table 2.1 *Attempts to Break Modified Caesar Cipher Text Using All Possibilities*

Cipher text	*K*	*W*	*U*	*M*	*P*	*M*	*Z*	*M*
Attempt Number (Value of k)								
1	L	X	V	N	Q	N	A	N
2	M	Y	W	O	R	O	B	O
3	N	Z	X	P	S	P	C	P
4	O	A	Y	Q	T	Q	D	Q
5	P	B	Z	R	U	R	E	R
6	Q	C	A	S	V	S	F	S
7	R	D	B	T	W	T	G	T
8	S	E	C	U	X	U	H	U
9	T	F	D	V	Y	V	I	V
10	U	G	E	W	Z	W	J	W
11	V	H	F	X	A	X	K	X
12	W	I	G	Y	B	Y	L	Y
13	X	J	H	Z	C	Z	M	Z
14	Y	K	I	A	D	A	N	A
15	Z	L	J	B	E	B	O	B
16	A	M	K	C	F	C	P	C
17	B	N	L	D	G	D	Q	D
18	C	O	M	E	H	E	R	E
19	D	P	N	F	I	F	S	F
20	E	Q	O	G	J	G	T	G
21	F	R	P	H	K	H	U	H
22	G	S	Q	I	L	I	V	I
23	H	T	R	J	M	J	W	J
24	I	U	S	K	N	K	X	K
25	J	V	T	L	O	L	Y	L

To put it mathematically, we can now have any permutation or combination of the 26 alphabets, which means (26 x 25 x 24 x 23 x … 2) or 4 x 1026 possibilities! This is extremely hard to crack. It might actually take years to try out these many combinations even with the most modern computers.

Mono-alphabetic ciphers pose a difficult problem for a cryptanalyst because it can be very difficult to crack thanks to the high number of possible permutations and combinations.

There is only one hitch. If the cipher text created with this technique is short, the cryptanalyst can try different attacks based on her knowledge of the English language. As we know, some alphabets in the English language occur more frequently than others. Language analysts have found that given a single alphabet in cipher text, the probability that it is a *P* is 13.33% - the highest. After P comes Z, which is likely to occur 11.67%. The probability that the alphabet is *C, K, L, N* or *R* is almost 0 – the lowest.

A cryptanalyst looks for patterns of alphabets in a cipher text, substitutes the various available alphabets in place of cipher text alphabets and then tries her attacks.

Apart from single-alphabet replacements, the cryptanalyst also looks for repeated patterns of words to try the attacks. For example, the cryptanalyst might look for two-alphabet cipher text patterns since the word *to* occurs very frequently in English. If the cryptanalyst finds that two alphabet combinations are found frequently in a cipher text message, she might try and replace all of them with *to* and then try and deduce the remaining alphabets/words. Next, the cryptanalyst might try to find repeating three-alphabet patterns and try and replace them with the word *the* and so on.

Playfair Cipher

The **Playfair Cipher**, also called as **Playfair Square**, is a cryptographic technique that is used for manual encryption of data. This scheme was invented by Charles Wheatstone in 1854. However, eventually the scheme came to be known by the name of Lord Playfair, who was Wheatstone's friend.

Table 2.2 *Vigenère Tableau*

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

Playfair made this scheme popular and hence his name was used. The Playfair Cipher was used by the British army in World War I and by the Australians in World War II. This was possible because Playfair is quite fast to use and does not demand any special equipment to be used. It was used to protect important but not very

critical information, so that by the time the cryptanalysts could break it, the value of the information was nullified anyway! In today's world, of course, Playfair would be deemed as an outdated cryptographic algorithm and rightly so. Playfair now has only academic purpose except its usage in some crosswords that appear in several newspapers.The Playfair encryption scheme uses two main processes, as shown in Fig. 2.14.

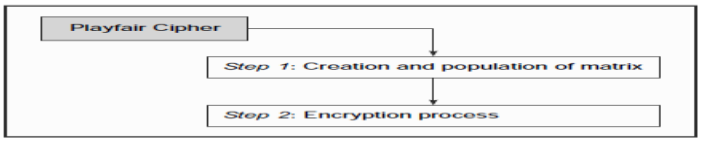


Fig. 2.14 *Playfair Cipher steps*

Step 1: Creation and Population of Matrix The Playfair Cipher makes use of a 5 x 5 matrix (table), which is used to store a *keyword* or *phrase* that becomes the *key* for encryption and decryption.The way this is entered into the 5 x 5 matrix is based on some simple rules, as shown in Fig. 2.15.

1. Enter the keyword in the matrix row-wise: left-to-right, and then top-to-bottom.
2. Drop duplicate letters.
3. Fill the remaining spaces in the matrix with the rest of the English alphabets (A-Z) that were not a part of our keyword. While doing so, combine I and J in the same cell of the table. In other words, if I or J is a part of the keyword, disregard both I and J while filling the remaining slots.

Fig. 2.15 *Matrix creation and population*

For example, suppose that our keyword is PLAYFAIR EXAMPLE. Then, the 5 x 5 matrix containing our keyword will look as shown in Fig. 2.16.

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

Fig. 2.16 *Keyword matrix for our example*

Hill Cipher is works on multiple letters at the same time. Hence, it is a type of Polygraphic Substitution Cipher. Lester Hill invented this in 1929. Hill cipher has its roots in matrix theory of Mathematics. More specifically, we need to know how to compute the inverse of a matrix. This mathematics is explained in *Appendix A.* Interested readers are encouraged to refer to the mathematical theory there.

The way the Hill Cipher works is as shown in Fig. 2.31.

1. Treat every letter in the plain text message as a number, so that A = 0, B = 1, …, Z = 25.
2. The plain text message is organized as a matrix of numbers, based on the above conversion. For example, if our plain text is CAT. Based on the above step, we know that C = 2, A = 0, and T = 19. Therefore, our plain text matrix would look as follows:

$$\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$$

3. Now, our plain text matrix is multiplied by a matrix of randomly chosen keys. The key matrix consists of size n x n, where n is the number of rows in our plain text matrix. For example, we take the following key matrix:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

4. Now multiply the two matrices, as shown below:

$$\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \times \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} = \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix}$$

5. Now compute a *mod 26* value of the above matrix. That is, take the remainder after dividing the above matrix values by 26. That is:

$$\begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \quad \text{mod } 26 = \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix}$$

6. (This is because: 31 / 26 = 1 with a remainder of 5: which goes in the above matrix, and so on).
7. Now, translating the numbers to alphabets, 5 = F, 8 = I, and 13 = N. Therefore, our cipher text is *FIN*.
8. For decryption, take the cipher text matrix and multiply it by the inverse of our original key matrix (explained later). The inverse of our original key matrix is:

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$$

Fig. 2.31(a) *Hill cipher example – Part 1/2*

The Hill Cipher is vulnerable to the *Known plain text attack*, which are going to discuss later. This is because it is linear (i.e. it is possible to computer smaller factors of the matrices, work on them individually and then join them back as and when they are ready).

1. For decryption, take the cipher text matrix and multiply it by the inverse of our original key matrix. The inverse of our original key matrix is:

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \times \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} = \begin{pmatrix} 210 \\ 442 \\ 305 \end{pmatrix}$$

2. Now we need to take modulo 26 of this matrix, as follows.

$$\begin{pmatrix} 210 \\ 442 \\ 305 \end{pmatrix} \bmod 26 = \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$$

3. Thus, our plain text matrix contains 2, 0, 19; which corresponds to 2 = C, 0 = A, and 19 = T. This gives is the original plain text back successfully.

Fig. 2.31(b) *Hill cipher example – Part 2/2*

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

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in *Step* 1 as a sequence of rows.

Fig. 2.32 *Rail fence technique*

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.

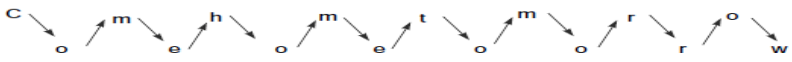
- ➢ plaintext = meet at the school house Key = 4 3 1 2 5 6 7

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.

Original plain text message: ***Come home tomorrow***

1. After we arrange the plain text message as a sequence of diagonals, it would look as follows (write the first character on the first line i.e. *C*, then second character on the second line, i.e. *o*, then the third character on the first line, i.e. *m*, then the fourth character on the second line, i.e. *e*, and so on). This creates a zigzag sequence, as shown below.



2. Now read the text row-by-row, and write it sequentially. Thus, we have: ***Cmhmtmrooeoeoorw*** as the cipher text.

Fig. 2.33 *Example of Rail fence technique*

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.

Drawbacks of Steganography

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

UNIT - II

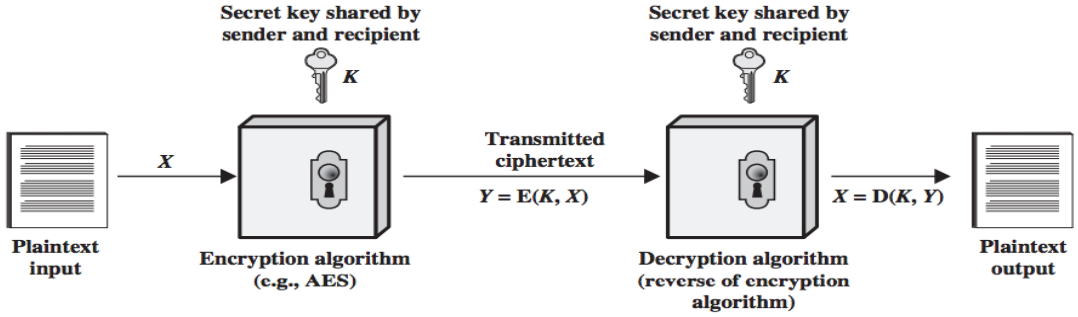
Symmetric key Ciphers: Block Cipher principles & Algorithms(DES, AES), Block cipher modes of operation, Stream ciphers, RC4,Location and placement of encryption function, Key distribution **Asymmetric key Ciphers:** Principles of public key cryptosystems, Algorithms(RSA, Diffie-Hellman), Key Distribution.

There are various transformation of plain text messages into cipher text messages. We have also seen the various methods for performing these transformations

CONVENTIONAL ENCRYPTION PRINCIPLES

A Conventional/Symmetric encryption scheme has five ingredients

1. ***Plain Text***: This is the original message or data which is fed into the algorithm asinput.
2. ***Encryption Algorithm***: This encryption algorithm performs various substitutions and transformations on the plain text.
3. ***Secret Key***: The key is another input to the algorithm. The substitutions and transformations performed by algorithm depend on the key.
4. ***Cipher Text*:** This is the scrambled (unreadable) message which is output of the encryption algorithm. This cipher text is dependent on plaintext and secret key. For a given plaintext, two different keys produce two different cipher texts.
5. ***Decryption Algorithm***: This is the reverse of encryption algorithm. It takes the cipher text and secret key as inputs and outputs the plain text.



The important point is that the security of conventional encryption depends on the secrecy of the key, not the secrecy of the algorithm i.e. it is not necessary to keep the algorithm secret, but only the key is to be kept secret. This feature that algorithm need

The generation of cipher text from plain text itself can be done in two basic ways, **stream ciphers** and **block ciphers**. This is shown in Fig. 3.1.

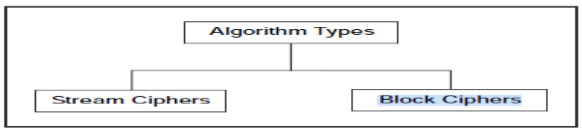


Fig. 3.1 *Types of cipher*

Stream Ciphers In **Stream Ciphers**, the plain text is encrypted one byte at a time. Suppose the original message (plain text) is *Pay 100* in ASCII (i.e. text format). When we convert these ASCII characters to their binary values, let us assume that it translate to 01011100 (hypothetically, just for simplicity, in reality, the binary text would be much larger as each text character takes seven bits). Suppose the key to be applied is 10010101 in binary.

Let us also assume that we apply the *XOR* logic as the encryption algorithm. XOR is quite simple to understand as shown in Fig. 3.2.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 3.2 *Functioning of XOR logic*

In simple terms, XOR produces an output of 1 only if one input is 0 and the other is 1. The output is 0 if both the inputs are 0 or if both the inputs are 1 (hence the name *exclusive*).
We can see the effect of XOR in Fig. 3.3.

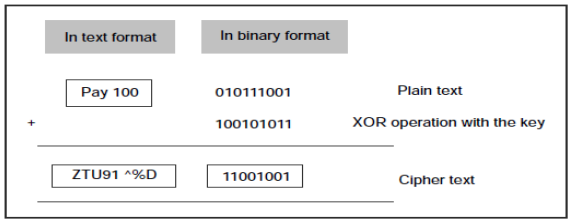


Fig. 3.3 *Stream ciphers*

As a result of applying one bit of key for every respective bit of the original message, suppose the cipher text is generated as 11001001 in binary (ZTU91 ^% in text). Note that each byte of the plain text is encrypted one after the other. Thus, what is transmitted is 11001001 in binary, which even when translated back to ASCII would mean ZTU91 ^%. This makes no sense to an attacker and thus protects the information.

Stream Cipher technique involves the encryption of one plain text byte at a time. The decryption also happens one byte at a time.

Another interesting property of XOR is that when used twice, it produces the original data. For example, suppose we have two binary numbers A = 101 and B = 110. We now want to perform an XOR operation on A and B to produce a third number C, i.e.:

C = A XOR B

Thus, we will have:

C = 101 XOR 110
= 011

Now, if we perform C XOR A, we will get B. That is:

B = 011 XOR 101
= 110

Similarly, if we perform C XOR B, we will get A. That is:

A = 011 XOR 110
= 101

This reversibility of XOR operations has many implications in cryptographic algorithms, as we shall notice.

XOR is reversible – when used twice, it produces the original values. This is useful in cryptography.

Block Ciphers

In **Block Ciphers**, rather than encrypting one byte at a time, a block of bytes is encrypted at one go. Suppose we have a plain text *FOUR_AND_FOUR* that needs to be encrypted. Using block cipher, *FOUR* could be encrypted first, followed by *_AND_* and finally *FOUR*. Thus, one block of characters gets encrypted at a time. During decryption, each block would be translated back to the original form. In actual practice, the communication takes place only in bits. Therefore, *FOUR* actually means binary equivalent of the ASCII characters *FOUR*. After any algorithm encrypts these, the resultant bits are converted back into their ASCII equivalents. Therefore, we get funny symbols such as *Vfa%*, etc. In actual practice, their binary equivalents are received, which are decrypted back into binary equivalent of ASCII *FOUR*. This is shown in Fig. 3.4.

An obvious problem with block ciphers is repeating text. For repeating text patterns, the same cipher is generated. Therefore, it could give a clue to the cryptanalyst regarding the original plain text. The cryptanalyst can look for repeating strings and try to break them. If she succeeds in doing so, there is a danger that a relatively large portion of the original message is broken into and therefore, the entire message can then be revealed with more effort. Even if the cryptanalyst cannot guess the remaining words, suppose she changes all *debit* to *credit* and vice versa in a funds transfer message, it could cause havoc! To deal with this problem, block ciphers are used in **chaining mode**, as we shall study later. In this approach, the previous block of cipher text is mixed with the current block, so as to obscure the cipher text, thus avoiding repeated patterns of blocks with the same content.

Block Cipher technique involves encryption of one block of text at a time. Decryption also takes one block of encrypted text at a time.

Practically, the blocks used in block cipher generally contain 64 bits or more. As we have seen, stream ciphers encrypt one byte at a time. This can be very time consuming and actually unnecessary in real life. That is why block ciphers are used more often in computer-based cryptographic algorithms as compared to stream ciphers. Consequently, we will focus our attention on block ciphers with reference to algorithm modes. However, as we shall see, two of the block cipher algorithm modes can also be implemented as stream cipher modes.

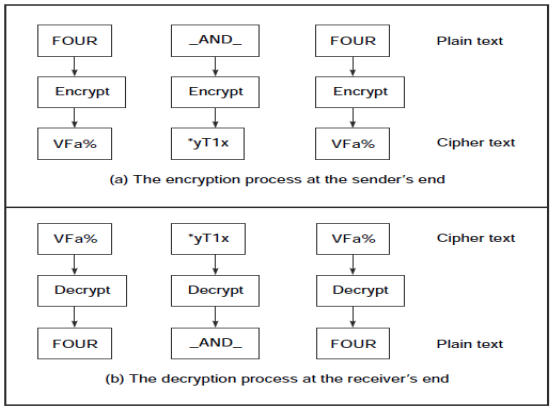


Fig. 3.4 *Block ciphers*

Data Encryption Standard (DES):

The **Data Encryption Standard (DES)**, also called as the Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO, has been a cryptographic algorithm used for over three decades. Of late, DES has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been slightly on the decline. However, no book on security is complete without DES, as it has been a landmark in cryptographic algorithms. We shall also discuss DES at length to achieve two objectives:

Firstly to learn about DES, but secondly and more importantly, to dissect and understand a real-life cryptographic algorithm. Using this philosophy, we shall then discuss some other cryptographic algorithms, but only at a conceptual level; because the in-depth discussion of DES would have already helped us understand in depth, how computer-based cryptographic algorithms work. DES is generally used in the ECB, CBC or the CFB mode.

The origins of DES go back to 1972, when in the US, the National Bureau of Standards (NBS), now known as the National Institute of Standards and Technology (NIST) embarked upon a project for protecting the data in computers and computer communications. They wanted to develop a single cryptographic algorithm. After two years, NBS realized that IBM's **Lucifer** could be considered as a serious candidate, rather than developing a fresh algorithm from scratch. After a few discussions, in 1975, the NBS published the details of the algorithm. Towards the end of 1976, the US Federal Government decided to adopt this algorithm and soon, it was renamed as *Data Encryption Standard (DES)*. Soon, other bodies also recognized and adopted DES as a cryptographic algorithm

How DES Works

Basic Principles DES is a block cipher. It encrypts data in blocks of size 64 bits each. That is, 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is shown in Fig. 3.18

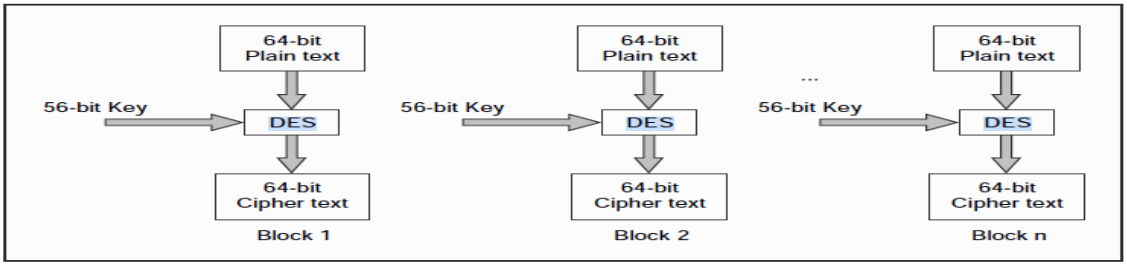


Fig. 3.18 *The conceptual working of DES*

We have mentioned that DES uses a 56-bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every eighth bit of the key is discarded to produce a 56-bit key. That is, bit positions 8, 16, 24, 32, 40, 48, 56 and 64 are discarded. This is shown in Fig. 3.19 with shaded bit positions indicating

discarded bits. (Before discarding, these bits can be used for parity checking to ensure that the key does not contain any errors.)

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

Fig. 3.19 *Discarding of every 8th bit of the original key (Shaded bit positions are discarded)*

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key, as shown in Fig. 3.20.

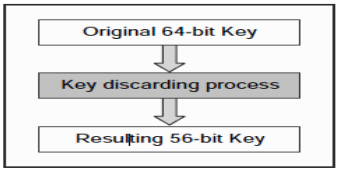


Fig. 3.20 *Key discarding process*

OVERALL STRUCTURE

Figure below shows the sequence of events that occur during an encryption operation. DES performs an initial permutation on the entire 64 bit block of data. It is then split into 2, 32 bit sub-blocks, Li and Ri which are then passed into what isknown as a round (see figure 2.3), of which there are 16 (the subscript i in Li and Ri indicates the current round). Each of the rounds are identical and the effects of increasing their number is twofold - the algorithms security is increased and its temporal efficiency decreased. Clearly these are two conflicting outcomes and a compromise must be made. For DES the number chosen was 16, probably to guarantee the elimination of any correlation between the ciphertext and either the plaintext or key6 . At the end of the 16th round, the 32 bit Li and Ri output quantities are swapped to create what is known as the pre-output. This [R16, L16] concatenation is permuted using a function which is the exact inverse of the initial permutation. The output of this final permutation is the 64 bit ciphertext.

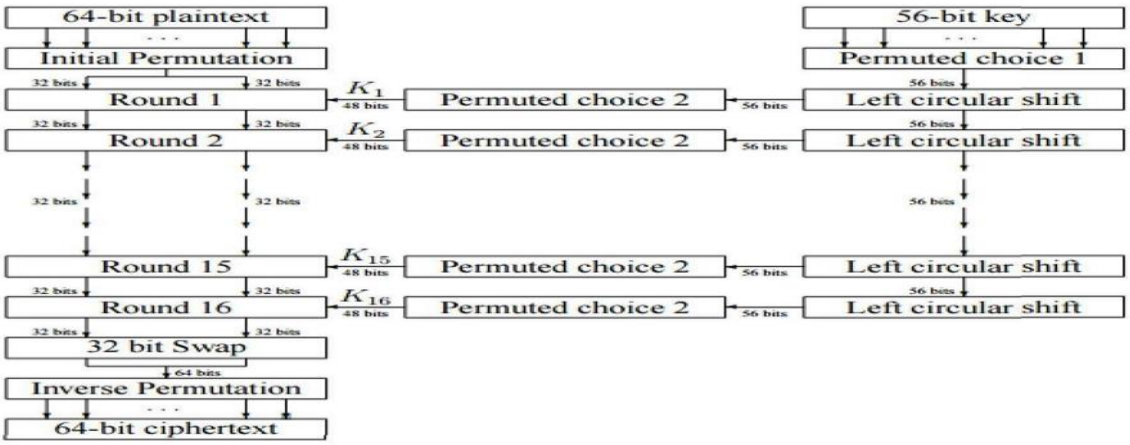


Figure : Flow Diagram of DES algorithm for encrypting data.

Initial Permutation (IP) As we have noted, the Initial Permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in Fig. 3.22. For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text block, the second bit with the 50th bit of the original plain text block and so on. This is nothing but jugglery of bit positions of the original plain text block.

Bit position in the plain text block	To be overwritten with the contents of this bit position
1	58
2	60
3	42
..	..
64	7

Fig. 3.22 *Idea of IP*

The complete transposition table used by IP is shown in Fig. 3.23. This table (and all others in this chapter) should be read from left to right, top to bottom. For instance, we have noted that 58 in the first position indicate that the contents of the 58th bit in the original plain text block will overwrite the contents of the 1st bit position, during IP.

Similarly, 1 is shown at the 40th position in the table, which means that the first bit will overwrite the 40th bit in the original plain text block. The same rule applies for all the other bit positions.

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

Fig. 3.23 *Initial Permutation (IP) table*

As we have noted, after IP is done, the resulting 64-bit permuted text block is divided into two half blocks. Each half block consists of 32 bits. We have called the left block as LPT and the right block as RPT. Now, 16 *rounds* are performed on these two blocks. This process is described as follows.

Rounds Each of the 16 rounds, in turn, consists of the broad level steps outlined in Fig. 3.24.

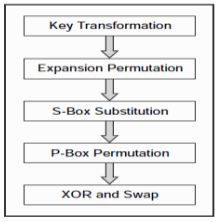


Fig. 3.24 *Details of one round in DES*

Step 1: Key transformation We have noted that the initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each round, a 56-bit key is available. From this 56- it key, a different 48-bit **sub-key** is generated during each *round* using a process called as **key transformation**. For this, the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round. For example, if the *round* number is 1, 2, 9 or 16, the shift is done by only position. For other rounds, the circular shift is done by two positions. The number of key bits shifted per round is shown in Fig. 3.25.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Fig. 3.25 *Number of key bits shifted per round*

After an appropriate shift, 48 of the 56 bits are selected. For selecting 48 of the 56 bits, the table shown in Fig. 3.26 is used. For instance, after the shift, bit number 14 moves into the first position, bit number 17 moves into the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce the 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub-set of the original 56-bit key, it is called as **compression permutation**.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Fig. 3.26 *Compression permutation*

Because of this compression permutation technique, a different subset of key bits is used in each round. That makes DES not so easy to crack.

Step 2: Expansion permutation Recall that after Initial Permutation, we had two 32-bit plain text areas, called as Left Plain Text (LPT) and Right Plain Text (RPT). During **expansion permutation**, the RPT is expanded from 32 bits to 48 bits. Besides increasing the bit size from 32 to 48, the bits are permuted as well, hence the name *expansion permutation*. This happens as follows:

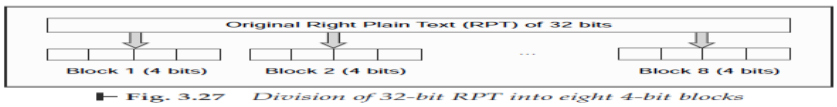


Fig. 3.27 *Division of 32-bit RPT into eight 4-bit blocks*

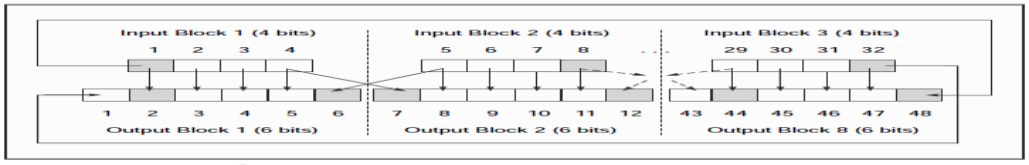


Fig. 3.28 *RPT expansion permutation process*

Step 3: S-box substitution **S-box substitution** is a process that accepts the 48-bit input from the XOR operation involving the compressed key and expanded RPT and produces a 32-bit output using the

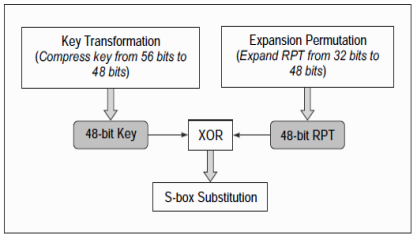


Fig. 3.30 *Way to S-box substitution*

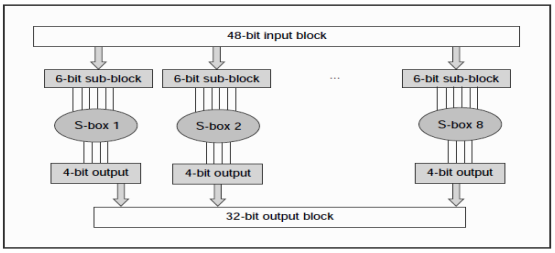


Fig. 3.31 *S-box substitution*

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

Fig. 3.32 *(a) S-box 1*

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

Fig. 3.32 *(b) S-box 2*

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

Fig. 3.32 *(c) S-box 3*

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

Fig. 3.32 *(d) S-box 4*

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

Fig. 3.32 *(e) S-box 5*

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

Fig. 3.32 *(f) S-box 6*

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

Fig. 3.32 *(g) S-box 7*

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

Fig. 3.32 *(h) S-box 8*

ADVANCED ENCRYPTION ALGORITHM (AES)

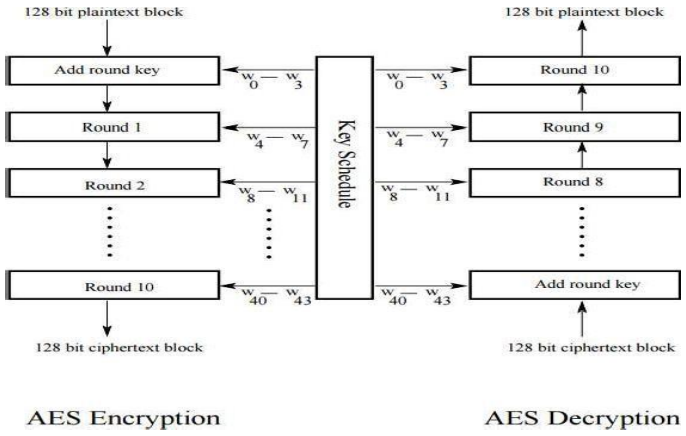
- AES is a block cipher with a block length of 128 bits. ˆ
- AES allows for three different key lengths: 128, 192, or 256 bits.
- Most of our discussion will assume that the key length is 128 bits.
- With regard to using a key length other than 128 bits, the main thing that changes in AES is how you generate the key schedule from the key — an issue I address at the end of Section 8.8.1. The notion of key schedule in AES
- Encryption consists of 10 rounds of processing for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. ˆ
- Except for the last round in each case, all other rounds are identical. ˆ
- Each round of processing includes one single-byte based substitution step, a row-wise permutation step, a column-wise mixing step, and the addition of the round key. The order in which these four steps are executed is different for encryption and decryption.

To appreciate the use of "row" and "column" in the previous bullet, you need to think of the input 128-bit block as consisting of a 4 × 4 array of bytes, arranged as follows

$$\begin{bmatrix} byte_0 & byte_4 & byte_8 & byte_{12} \\ byte_1 & byte_5 & byte_9 & byte_{13} \\ byte_2 & byte_6 & byte_{10} & byte_{14} \\ byte_3 & byte_7 & byte_{11} & byte_{15} \end{bmatrix}$$

Therefore, the first four bytes of a 128-bit input block occupy the first column in the 4 × 4 matrix of bytes. The next four bytes occupy the second column, and so on.

The 4×4 matrix of bytes shown above is referred to as the state array in AES



Block cipher modes of operation:

An *algorithm mode* is a combination of a series of the basic algorithm steps on block cipher and some kind of feedback from the previous step. We shall discuss it now, as it forms the basis for the computer- based security algorithms. There are four important algorithm modes, namely, **Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB)** and **Output Feedback (OFB)**. This is shown in Fig. 3.5. The first two modes operate on block cipher, whereas the latter two modes are block cipher modes, which can be used as if they are working on stream cipher.

Apart from this, we also discuss a variation of the OFB mode, called as **Counter (CTR)**.

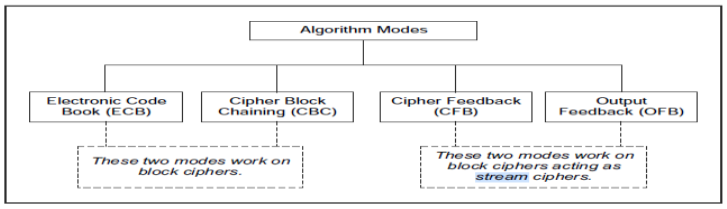


Fig. 3.5 *Algorithm modes*

Electronic Code Book (ECB) Mode

Electronic Code Book (ECB) is the simplest mode of operation. Here, the incoming plain text message is divided into blocks of 64 bits each. Each such block is then encrypted independently of the other blocks. For all blocks in a message, the same key is used for encryption. This process is shown in Fig. 3.6.

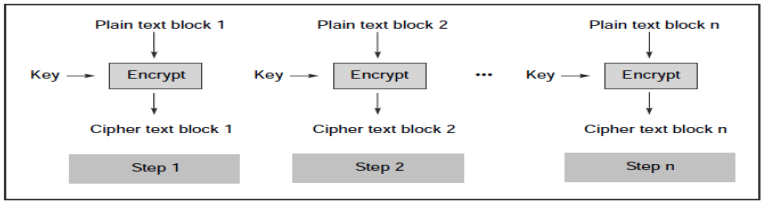


Fig. 3.6 *ECB mode – The encryption process*

At the receiver's end, the incoming data is divided into 64-bit blocks and by using the same key as was used for encryption, each block is decrypted to produce the corresponding plain text block. This process is shown in Fig. 3.7.

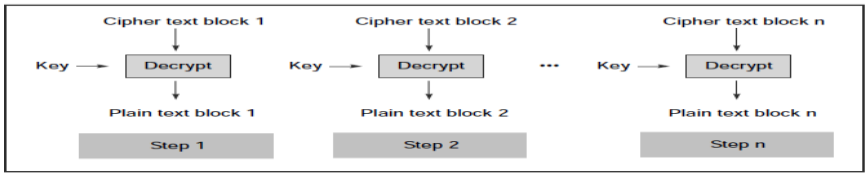


Fig. 3.7 *ECB mode – The decryption process*

- ✓ In ECB, since a single key is used for encrypting all the blocks of a message, if a plain text block repeats in the original message, the corresponding cipher text block will also repeat in the encrypted message. Therefore, ECB is suitable only for encrypting small messages, where the scope for repeating the same plain text block is quite less.

Cipher Block Chaining (CBC) Mode

We saw that in the case of ECB, within a given message (i.e. for a given key), a plain text block always produces the same cipher text block. Thus, if a block of plain text occurs more than once in the input, the corresponding cipher text block will also occur more than once in the output, thus providing some clues to a cryptanalyst. To overcome this problem, the **Cipher Block Chaining (CBC)** mode ensures that even if a block of plain text repeats in the input, these two (or more) identical plain text blocks yield totally different cipher text blocks in the output. For this, a feedback mechanism is used, as we shall learn in the following paragraphs.

Chaining adds a feedback mechanism to a block cipher. In Cipher Block Chaining (CBC), the results of the encryption of the previous block are fed back into the encryption of the current block. That is, each block is used to modify the encryption of the next block. Thus, each block of cipher text is dependent on the corresponding current input plain text block, as well as all the previous plain text blocks.

The encryption process of CBC is depicted in Fig. 3.8 and described thereafter.

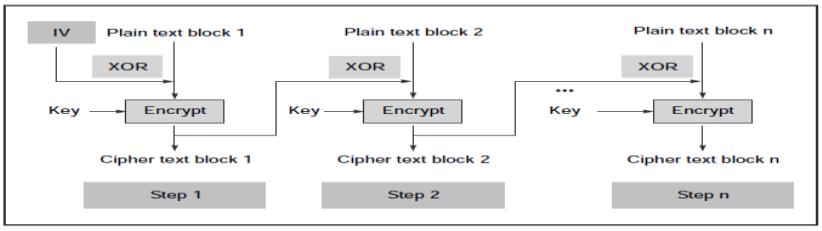


Fig. 3.8 *CBC mode – The encryption process*

1. As shown in the figure, the first step receives two inputs: the first block of plain text and a random block of text, called as **Initialization Vector (IV)**.
   a. The IV has no special meaning: it is simply used to make each message unique. Since the value of IV is randomly generated, the likelihood of it repeating in two different messages is quite rare. Consequently, IV helps in making the cipher text somewhat unique or at least quite different from all the other cipher texts in a different message. Interestingly, it is not mandatory to keep IV secret – it can be known to everybody. This seems slightly concerning and confusing. However, if we re-look at the operation of CBC, we will realize that IV is simply one of the two inputs to the first encryption step. The output of Step 1 is Cipher text block 1, which is also one of the two inputs to the second encryption step. In other words, Cipher text block 1 is also an IV for Step 2! Similarly, Cipher text block 2 is also an IV for Step 3 and so on. Since all these cipher text blocks will be sent to the receiver, we are actually anyway sending all IVs for Step 2 onwards! Thus, there is no special reason why the IV for Step 1 should be kept secret. It is the key used for encryption is what needs to be kept secret. However, in practice, for maximum security, both the key and the IV are kept secret.
   b. The first block of cipher text and IV are combined using XOR and then encrypted using a key to produce the first cipher text block. The first cipher text block is then provided as a *feedback* to the next plain text block, as explained below.
2. In the second step, the second plain text block is XORed with the output of Step 1, i.e. the first cipher text block. It is then encrypted with the same key, as used in Step 1. This produces cipher text block 2.
3. In the third step, the third plain text block is XORed with the output of Step 2, i.e. the second cipher text block. It is then encrypted with the same key, as used in Step 1.
4. This process continues for all the remaining plain text blocks of the original message. Remember that the IV is used only in the first plain text block. However, the same key is used for encryption of all plain text blocks.

The decryption process works as follows.

1. The Cipher text block 1 is passed through the decryption algorithm using the same key, which was used during the encryption process for all the plain text blocks. The output of this step is then XORed with the IV. This process yields Plain text block 1.
2. In Step 2, the Cipher text block 2 is decrypted and its output is XORed with Cipher text block 1, which yields Plain text block 2.
3. This process continues for all the Cipher text blocks in the encrypted message. The decryption process is shown in Fig. 3.9.

Appendix A provides some more mathematical details about the CBC mode.

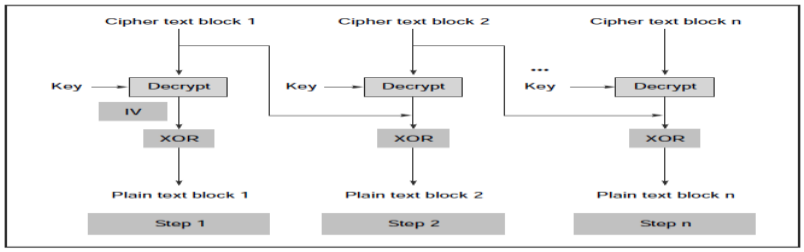


Fig. 3.9 *CBC mode – The decryption process*

Output Feedback (OFB) Mode The **Output Feedback (OFB)** mode is extremely similar to the CFB. The only difference is that in the case of CFB, the cipher text is fed into the next stage of encryption process. But in the case of OFB, the output of the IV encryption process is fed into the next stage of encryption process. Therefore, we shall not describe the details of OFB and instead, shall simply draw the block diagram of the OFB process, as shown in Fig. 3.14. The same details as discussed in CFB apply here, excepting the change, as pointed out above.

Let us summarize the key advantage of the OFB mode. In simple terms, we can state that in this mode, if there are errors in individual bits, they remain errors in individual bits and do not corrupt the whole message. That is, bit errors do not get propagated. If a cipher text bit Ci is in error, only the decrypted value corresponding to this bit, i.e. Pi is wrong. Other bits are not affected. Remember that in contrast to this, in the CFB mode, the cipher text bit Ci is fed back as input to the shift register and would corrupt the other bits in the message!

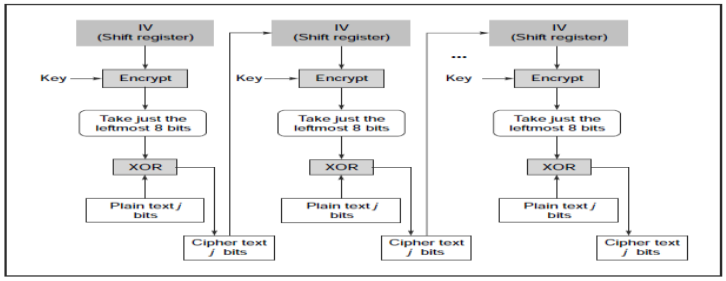


Fig. 3.13 *CFB – The overall encryption process*

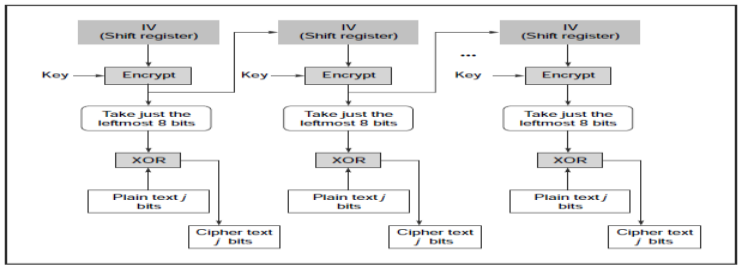


Fig. 3.14 *OFB – The overall encryption process*

Counter (CTR) Mode The **Counter (CTR)** mode is quite similar to the OFB mode, with one variation. It uses *sequence numbers* called as *counters* as the inputs to the algorithm. After each block is encrypted, to fill the register, the next counter value is used. Usually, a constant is used as the initial counter value and is incremented (usually by 1) for every iteration. The size of the counter block is the same as that of the plain text block.

For encryption, the counter is encrypted and then XORed with the plain text block to get the cipher text. No chaining process is used. On the other hand, for decryption, the same sequence of counters is used. Here, each encrypted counter is XORed with the corresponding cipher text block to obtain the original plain text block. The overall operation of the *Counter mode* is shown in Fig. 3.15 and Fig. 3.16.

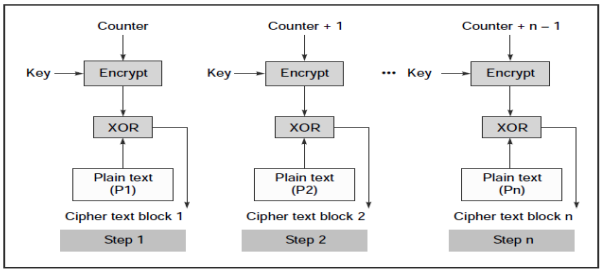


Fig. 3.15 *Counter (CTR) mode: The encryption process*

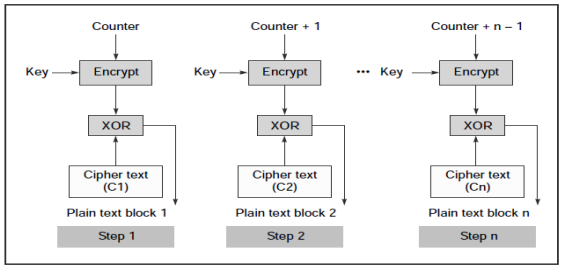


Fig. 3.16 *Counter (CTR) mode: The decryption process*

The encryption or decryption process in Counter can be done in parallel on multiple text blocks, since no chaining is involved. This can make the execution speed of Counter faster. Multiprocessing systems can take advantage of this feature to introduce features that help reduce the overall processing time. Preprocessing can be achieved to prepare the output of the encryption boxes that input to the XOR operations. The Counter mode mandates implementation of the encryption process only and not of the decryption process.

Table 3.1 summarizes the key features of the various algorithm modes.

Table 3.1 *Algorithm Modes: Details and Usage*

Algorithm mode	*Details*	*Usage*
Electronic Code Book (ECB)	The same key independently encrypts blocks of text, 64 bits at a time.	Transmitting a single value in a secure fashion (e.g. password or key used for encryption)
Cipher Block Chaining (CBC)	64 bits cipher text from the previous step and 64 bits plain text of the next step are XORed together.	Encrypting blocks of text Authentication
Cipher Feedback (CFB)	*K* bits of randomized cipher text from the previous step and *K* bits plain text of the next step are XORed together.	Transmitting encrypted stream of data Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption step is the preceding DES output.	Transmitting encrypted stream of data
Counter (CTR)	A counter and plain text block are encrypted together, after which the counter is incremented.	Block-oriented transmissions Applications needing high speed

RC4

RC4 was designed by Ron Rivest of RSA Security in 1987. The official name for this algorithm is "Rivest Cipher 4". However, because of its ease of reference, the acronym **RC4** has stuck.

RC4 is a stream cipher. This means that the encryption happens byte-by-byte. However, this can be changed to bit-by-bit encryption (or to a size other than a byte/bit).

RC4 was initially kept secret. However, in September 1994, a description of the algorithm was anonymously posted to the *Cypherpunks* mailing list. Thereafter, it was posted on the *sci.crypt* newsgroup and from there to many other Internet sites. Because the algorithm is now well known, it is no longer a secret. However, we should note that there is a trademark associated with the name "RC4".

An interesting remark says that "The current status seems to be that *unofficial* implementations are legal, but cannot use the RC4 name."

RC4 has become part of some widely used encryption techniques and standards, including Wireless Equivalent Privacy (WEP) and WPA for wireless cards and TLS. What has made its wide deployment possible is its speed and simplicity of design. Implementations in both software and hardware are possible. RC4 does not consume many resources

Description

RC4 generates a pseudorandom stream of bits called as **keystream**. This is combined with the plain text using XOR for encryption. Even decryption is performed in a similar manner.

Let us understand this in more detail in the following paragraphs..

There is a variable length key consisting of 1 to 256 bytes (or 8 to 2048 bits). This key is used to initialize a 256-byte *state vector* with elements identified as S[0].S[1]. ...., S[255].

To perform encryption or decryption operation, one of these 256 bytes of S is selected and processed. We will call the resulting output as *k.* After this, the entries in S are permuted once again. Overall, there are two processes involved: (a) Initialization of S and (b) Stream generation. We describe these as follows:

Initialization of S This process consists of the following steps.

1. Choose a key (K) of length between 1 and 256 bytes.
2. Set the values in the state vector S equal to the values from 0 to 255 in an ascending order. In other words, we should have S[0] = 0, S[1] = 1, …, S[255] = 255.

3. Create another temporary array T. If the length of the key K (termed as *keylen*) is 256 bytes, copy K into T as is. Otherwise, after copying K to T, whatever are the remaining positions in T are filled with the values of K again. At the end, T should be completely filled.

Thus, following steps are executed:

```
for i = 0 to 255
// Copy the current value of i into the current position in the S array
S [i] = i;

// Now copy the contents of the current position of the K array into T. If K is exhausted, loop back
// to get the values of the K array from the un-exhausted portion of K.
T [i] = K [i mod keylen];
```

A small Java program shown in Fig. 3.53 implements this logic. Here, we have considered that the K array contains 10 elements, i.e. *keylen* is 10.

```
public class InitRC4 {

        public static void main (String [] args) {

                int[] S, T, K;
                S = new int [255];
                T = new int [255];
                K = new int [255];

                int i;
                int keylen = 10;

                for (i=0; i<200; i++)
                        K [i] = i * 2;

                for (i=0; i<255; i++) {
                        S [i] = i;
                        T [i] = K [i % keylen];
                }
        }
}
```

Fig. 3.53 *Java code for implementing initialization of S Step*

After this, the T array is used to produce initial permutation of S. For this purpose, a loop executes, iterating i from 0 to 255. In each case, the byte at the position S[i] is swapped with another byte in the S array, as per an arrangement decided by T[i]. For this purpose, the following logic is used:

```
j = 0;

for i = 0 to 255
     j = (j + S [i] + T [i]) mod 256;
     swap (S [i], S [j]);
```

Note that this is just a permutation. The values of S are simply being rearranged, not changed. The corresponding Java portion is shown in Fig. 3.54.

Stream Generation Now that the S array is ready with the above initializations and permutations, the initial key array K is discarded. Now, we need to again loop for i = 0 to 255. In each step, we swap S

```
// Initial permutation of S

int j = 0, temp;

for (i=0; i<255; i++) {
        j = (j + S [i] + T [i]) % 256;
        temp = S [i];
        S [i] = T [i];
        T [i] = temp;
}

for (i=0; i<255; i++) {
        System.out.println ("S [i] = " + S [i]);
}
```

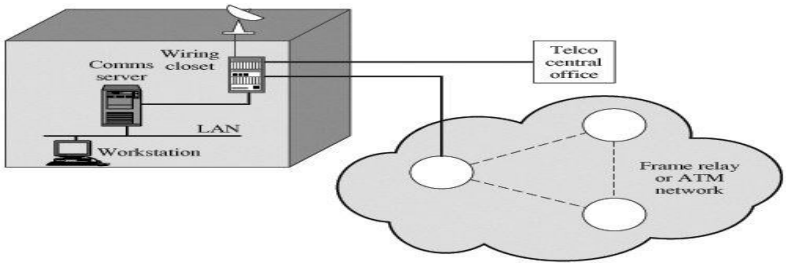
Fig. 3.54 *Java code for implementing permutation of S Step*

Location and placement of encryption function:

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

Potential Locations for Confidentiality Attacks

As an example, consider a user workstation in a typical business organization. Figure 7.1 suggests the types of communications facilities that might be employed by such a workstation and therefore gives an indication of the points of vulnerability.



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

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

Access to the outside world from the LAN is almost always available in the form of a router that connects to the Internet, a bank of dial-out modems, or some other type of communications server. From the communications server, there is a line leading to a wiring closet. The wiring closet serves as a patch panel for interconnecting internal data and phone lines and for providing a staging point for external communications.

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

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

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

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

An attack can take place on any of the communications links. For active attacks, the attacker needs to gain physical control of a portion of the link and be able to insert and capture transmissions. For a passive attack, the attacker merely needs to be able to observe transmissions. The communications links involved can be cable (telephone twisted pair, coaxial cable, or optical fiber), microwave links, or satellite channels.

Twisted pair and coaxial cable can be attacked using either invasive taps or inductive devices that monitor electromagnetic emanations. Invasive taps allow both active and passive attacks, whereas inductive taps are useful for passive attacks. Neither type of tap is as effective with optical fiber, which is one of the advantages of this medium. The fiber does not generate electromagnetic emanations and hence is not vulnerable to inductive taps. Physically breaking the cable seriously degrades signal quality and is therefore detectable. Microwave and satellite transmissions can be intercepted with little risk to the attacker. This is especially true of satellite transmissions, which cover a broad geographic area. Active attacks on microwave and satellite are also possible, although they are more difficult technically and can be quite expensive.

Key Distribution:

For symmetric encryption to work, the two parties to an exchange must share 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 attacker learns the key. Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link. However, for end-to-end encryption, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device needs a number of keys supplied dynamically. The problem is especially difficult in a wide area distributed system.

The scale of the problem depends on the number of communicating pairs that must be supported. If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate. Thus, if there are N hosts, the number of required keys is [N(N 1)]/2. If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication. Thus, a network may have hundreds of hosts but thousands of users and processes. Figure 7.7 illustrates the magnitude of the key distribution task for end-to-end encryption.[5] A network using node-level encryption with 1000 nodes would conceivably need to distribute as many as half a million keys. If that same network supported 10,000 applications, then as many as 50 million keys may be required for application-level encryption.

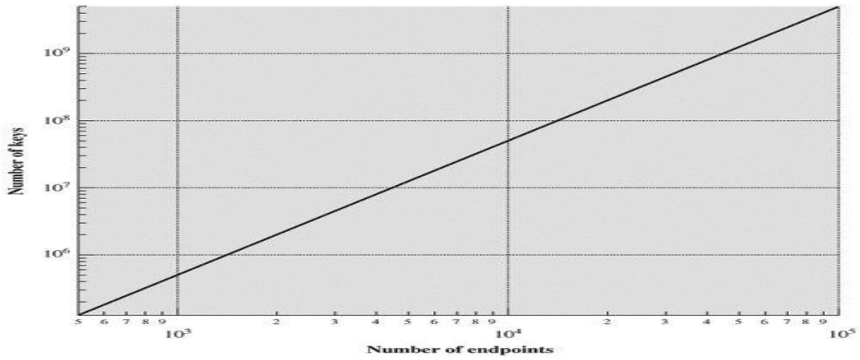


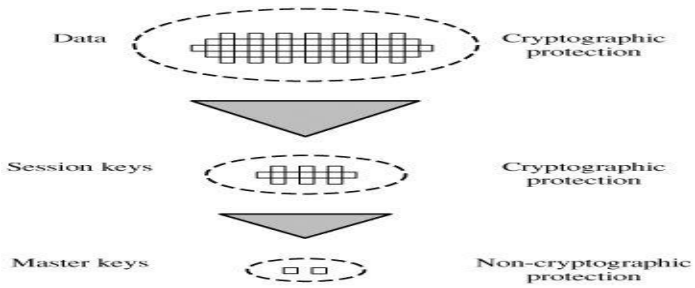
Figure 7.7. Number of Keys Required to Support Arbitrary Connections between Endpoints

Returning to our list, option 3 is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed. Furthermore, the initial distribution of potentially millions of keys must still be made.

For end-to-end encryption, some variation on option 4 has been widely adopted. In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed. Each user must share a unique key with the key distribution center for purposes of key distribution.

The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used (Figure 7.8). Communication between end systems is encrypted using a temporary key, often referred to as a **session key**. Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded. Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.

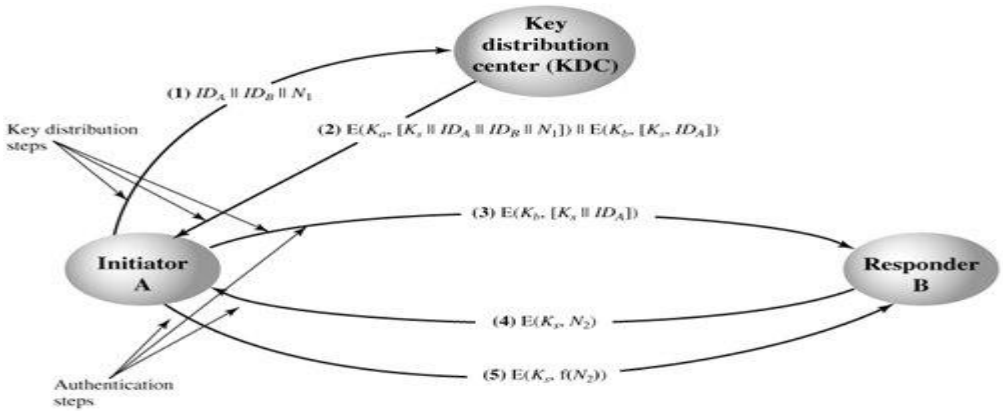
Figure 7.8. The Use of a Key Hierarchy



For each end system or user, there is a unique master key that it shares with the key distribution center. Of course, these master keys must be distributed in some fashion. However, the scale of the problem is vastly reduced. If there are N entities that wish to communicate in pairs, then, as was mentioned, as many as [N(N 1)]/2 session keys are needed at any one time. However, only N master keys are required, one for each entity. Thus, master keys can be distributed in some noncryptographic way, such as physical delivery.

A Key Distribution Scenario

The key distribution concept can be deployed in a number of ways. A typical scenario is illustrated in Figure 7.9, which is based on a figure in [POPE79]. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).



PUBLIC KEY CRYPTOGRAPHY

The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography. It is *asymmetric*, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key.

Public key schemes are neither more nor less secure than private key (security depends on the key size for both). Public-key cryptography *complements rather than replaces* symmetric cryptography. Both also have issues with key distribution, requiring the use
of some suitable protocol. The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

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

2.) ***digital signatures*** ~~h~~ow to verify a message comes intact from the claimed sender

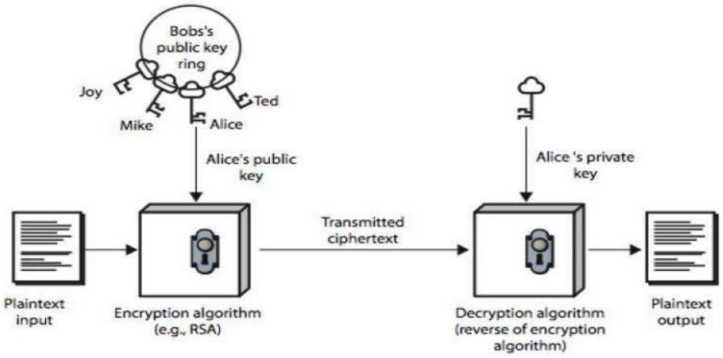
Public-key/two-key/asymmetric cryptography involves the use of **two** keys:

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

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

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

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



The essential steps are the following:

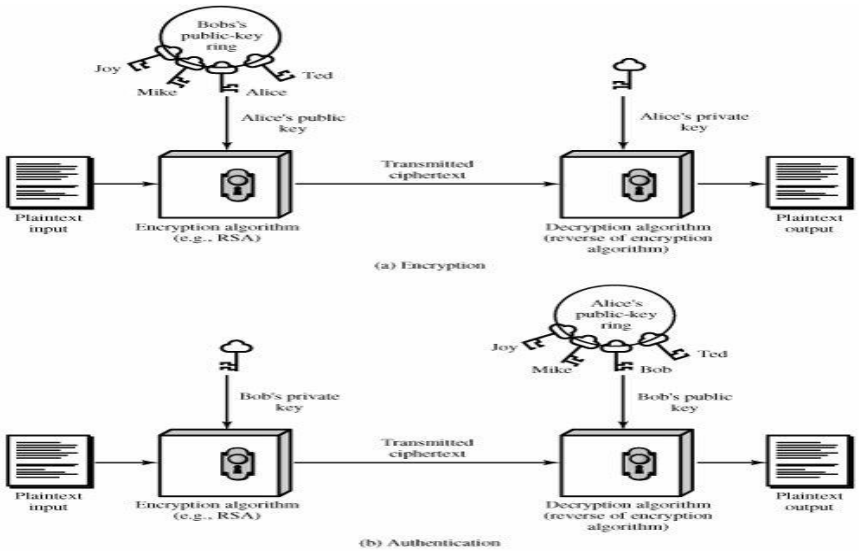
1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

Notations used in Public-key cryptography:

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

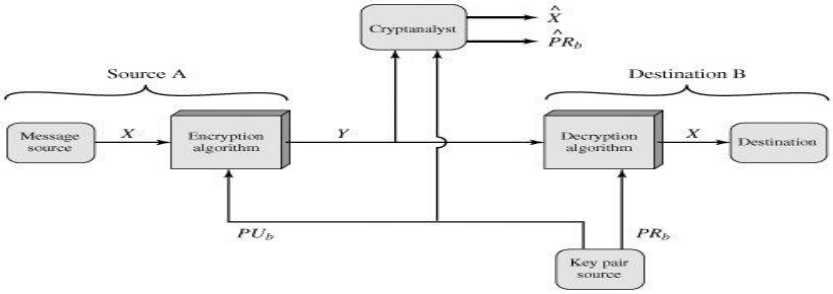
Table 9.1 summarizes some of the important aspects of symmetric and public-key encryption. To discriminate between the two, we refer to the key used in symmetric encryption as a secret key. The two keys used for asymmetric encryption are referred to as the **public key** and the **private key**.[2] Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption.

The following notation is used consistently throughout. A secret key is represented by K_m, where m is some modifier; for example, K_a is a secret key owned by user A. A public key is represented by PU_a, for user A, and the corresponding private key is PR_a, Encryption of plaintext X can be performed with a secret key, a public key, or a private key, denoted by $E(K_a, X)$, $E(PU_a, X)$, and $E(PR_a, X)$, respectively. Similarly, decryption of ciphertext C can be performed with a secret key, a public key, or a private key, denoted by $D(K_a, X)$, $D(PU_a, X)$, and $D(PR_a, X)$, respectively.



Let us take a closer look at the essential elements of a public-key encryption scheme, using Figure 9.2 (compare with Figure 2.2). There is some source A that produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M,]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, PU_b, and a private key, PU_b. PU_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

Figure 9.2. Public-Key Cryptosystem: Secrecy



Applications for Public-Key Cryptosystems

Before proceeding, we need to clarify one aspect of public-key cryptosystems that is otherwise likely to lead to confusion. Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories:

- Encryption/decryption: The sender encrypts a message with the recipient's public key.
- Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications. Table 9.2 indicates the applications supported by the algorithms discussed

Table 9.2. Applications for Public-Key Cryptosystems

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

The RSA Algorithm:

One of the first of the responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78].[4] The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

Apparently, the first workable public-key system for encryption/decryption was put forward by Clifford Cocks of Britain's CESG in 1973 [COCK73]; Cocks's method is virtually identical to RSA.

The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}. We examine RSA in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and cryptanalytical implications of RSA.

Description of the Algorithm

The scheme developed by Rivest, Shamir, and Adleman makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to $\log_2(n)$; in practice, the block size is i bits, where $2^i < n \leq 2^M$ and ciphertext block C:

$C = M^e \bmod n$

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

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PU = {d, n}. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that M^{ed} mod n = M for all M < n.
2. It is relatively easy to calculate mod M^e mod n and C^d for all values of M < n.
3. It is infeasible to determine d given e and n.

For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form

M^{ed} mod n = M

The preceding relationship holds if e and d are multiplicative inverses modulo f(n), where f(n) is the Euler totient function. It is shown in Chapter 8 that for p, q prime, f(pq) = (p 1)(q 1) The relationship between e and d can be expressed as Equation 9-1

$$ed \bmod \phi(n) = 1$$

This is equivalent to saying ed ≡ 1 mod f(n)

That is, e and d are multiplicative inverses mod f(n). Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to f(n). Equivalently, gcd(f(n),d) = 1. See Appendix 9A for a proof that Equation (9.1) satisfies the requirement for RSA.

We are now ready to state the RSA scheme. The ingredients are the following:

p,q, two prime numbers	(private, chosen)
n = pq	(public, calculated)
e, with gcd(f(n),e) = 1;1 < e < f(n)	(public, chosen)
d≡ f(n))	(private, calculated)

Key Generation

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

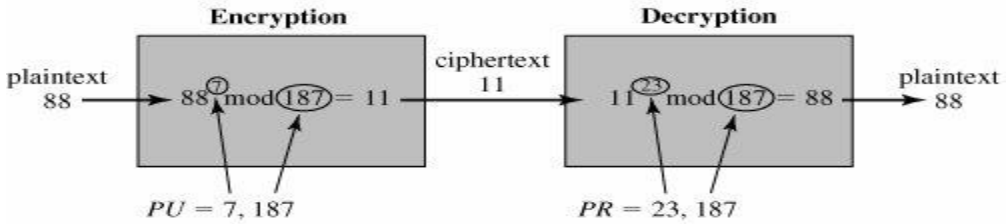
Encryption

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

Decryption

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

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17 x 11 = 187.
3. Calculate f(n) = (p 1)(q 1) = 16 x 10 = 160.
4. Select e such that e is relatively prime to f(n) = 160 and less than f(n) we choose e = 7.
5. Determine d such that de ≡ 1 (mod 160) and d = 23, because 23 x 7 = 161 = 10 x 160 + 1; d can be calculated using the extended Euclid's algorithm (Chapter 4).



The resulting keys are public key PU = {7,187} and private key PR = {23,187}. The example shows the use of these keys for a plaintext input of M = 88. For encryption, we need to calculate $C = 88^7$ mod 187. Exploiting the properties of modular arithmetic, we can do this as follows:

88^7 mod 187 = [(88^4 mod 187) x (88^2 mod 187) x (88^1 mod 187)] mod 187

88^1 mod 187 = 88

88^2 mod 187 = 7744 mod 187 = 77

88^4 mod 187 = 59,969,536 mod 187 = 132

88^7 mod 187 = (88 x 77 x 132) mod 187 = 894,432 mod 187 = 11

For decryption, we calculate $M = 11^{23}$ mod 187:

11^{23} mod 187 = [(11^1 mod 187) x (11^2 mod 187) x (11^4 mod 187) x (11^8 mod 187) x (11^8 mod 187)] mod 187

11^1 mod 187 = 11

11^2 mod 187 = 121

11^4 mod 187 = 14,641 mod 187 = 55

11^8 mod 187 = 214,358,881 mod 187 = 33
11^{23} mod 187 = (11 x 121 x 55 x 33 x 33) mod 187 = 79,720,245 mod 187 = 88

The Security of RSA

Four possible approaches to attacking the RSA algorithm are as follows:

- Brute force: This involves trying all possible private keys.
- Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.
- **Timing attacks**: These depend on the running time of the decryption algorithm.
- Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

Diffie-Hellman Key Exchange:

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76b] and is generally referred to as Diffie-Hellman key exchange.[1] A number of commercial products employ this key exchange technique.

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values.

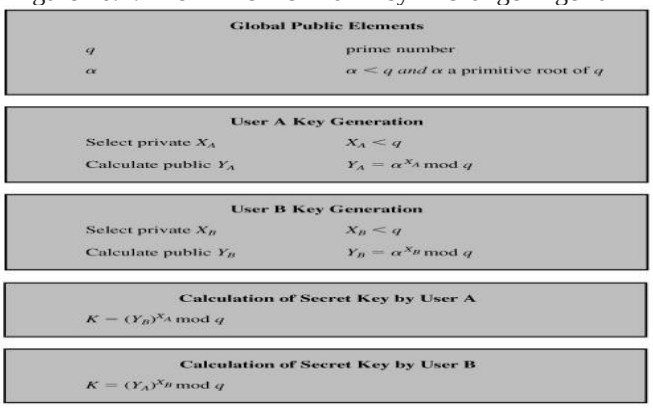
The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. First, we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to p 1. That is, if a is a primitive root of the prime number p, then the numbers a mod p, a^2 mod p,..., a^{p1} mod p are distinct and consist of the integers from 1 through p 1 in some permutation. For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that b ≡ i (mod p) where 0 ≤ i (The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as $dlog_{a,p}$ (b). See Chapter 8 for an extended discussion of discrete logarithms.

The Algorithm:

Figure 10.7 summarizes the Diffie-Hellman key exchange algorithm. For this scheme, there are two publicly known numbers: a prime number q and an integer that is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = a^{XA}$ mod q. Similarly, user B independently selects a random integer $X_A < q$ and computes $Y_B = a^{XB}$ mod q. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{X_A}$ mod q and user B computes the key as $K = (Y_A)^{XB}$ mod q. These two calculations produce identical results:

$K = (Y_B)^{XA}$ mod q
$= (a^{XB} \text{ mod } q)^{XA}$ mod q
$= (a^{XB})^{XA}$ mod q by the rules of modular arithmetic
$= (a^{XB\ XA}$ mod q
$= (a^{XA})^{XB}$ mod q
$= (a^{XA}$ mod q)
$= (a^{XA} \text{ mod } q)^{XB}$ mod q
$= (Y_A)^{XB}$ mod q

Figure 10.7. The Diffie-Hellman Key Exchange Algorithm



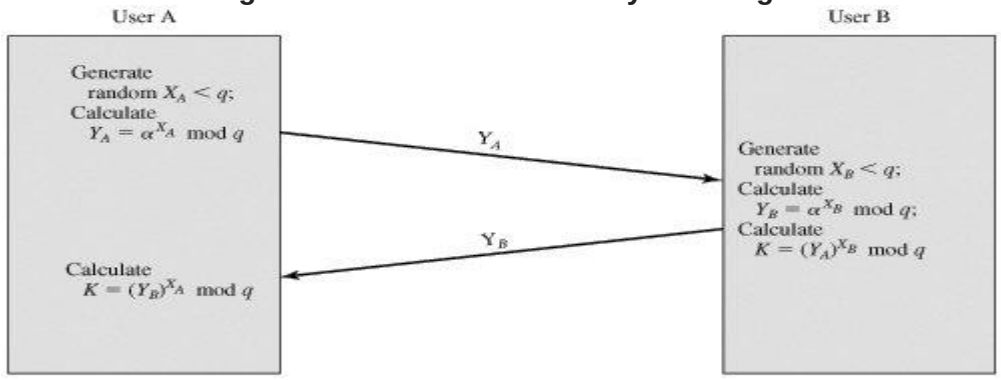
The result is that the two sides have exchanged a secret value. Furthermore, because X_A and X_B are private, an adversary only has the following ingredients to work with: q, a, Y_A, and Y_B. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

$$X_B = \text{dlog}_{\alpha,q}(Y_B)$$

The adversary can then calculate the key K in the same manner as user B calculates it. The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.
Here is an example. Key exchange is based on the use of the prime number q = 353 and a primitive root of 353, in this case a = 3. A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.
B computes $Y_B = 3^{233} \bmod 353 = 248$.
After they exchange public keys, each can compute the common secret key:
A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.
B computes $K = (Y_A)^{X_E} \bmod 353 = 40^{233} \bmod 353 = 160$.
We assume an attacker would have available the following information:
$q = 353$; $a = 3$; $Y_A = 40$; $Y_B = 248$
In this simple example, it would be possible by brute force to determine the secret key 160. In particular, an attacker E can determine the common key by discovering a solution to the equation $3^a \bmod 353 = 40$ or the equation $3^b \bmod 353 = 248$. The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provides $3^{97} \bmod 353 = 40$.
With larger numbers, the problem becomes impractical.
Key Exchange Protocols
Figure 10.8 shows a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A, calculate Y_A, and send that to user B. User B responds by generating a private value X_B calculating Y_B, and sending Y_B to user A. Both users can now calculate the key. The necessary public values q and a would need to be known ahead of time. Alternatively, user A could pick values for q and a and include those in the first message.

Figure 10.8. Diffie-Hellman Key Exchange



KEY MANAGEMENT

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

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

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

PUBLIC ANNOUNCEMENT OF PUBLIC KEYS

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

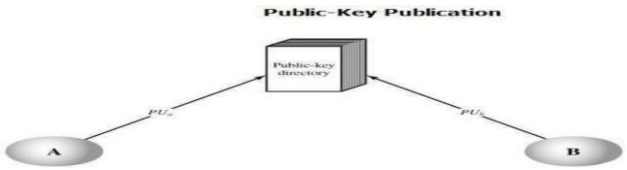
to messages that they send to public forums.

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

PUBLICLY AVAILABLE DIRECTORY

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

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



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

PUBLIC-KEY AUTHORITY

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

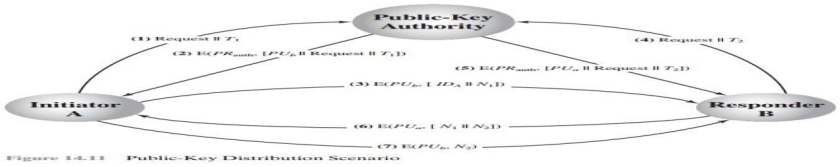


Figure 14.11 Public-Key Distribution Scenario

1. Alice sends a timestamped message to the central authority with a request for Bob's public key (the time stamp is to mark the moment of the request)
2. The authority sends back a message encrypted with its private key (for authentication) –message contains Bob's public key and the original message of Alice – this way Alice knows this is not a reply to an old request;
3. Alice starts the communication to Bob by sending him an encrypted message containing her identity IDA and a nonce N1 (to identify uniquely this transaction)
4. Bob requests Alice's public key in the same way (step 1)
5. Bob acquires Alice's public key in the same way as Alice did. (Step-2)
6. Bob replies to Alice by sending an encrypted message with N1 plus a new generated nonce N2 (to identify uniquely thetransaction)
7. Alice replies once more encrypting Bob's nonce N2 to assure bob that its correspondent is Alice
8. Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a technique known as caching.
9. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

PUBLIC-KEY CERTIFICATES

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

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

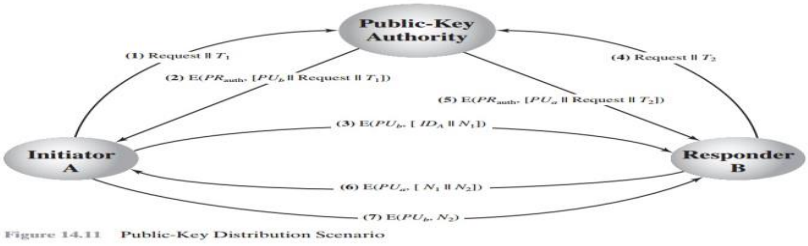
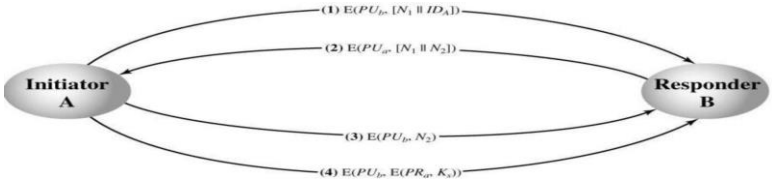


Figure 14.11 Public-Key Distribution Scenario

SECRET KEY DISTRIBUTION WITH CONFIDENTIALITY AND AUTHENTICATION:

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



1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N_1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (1), the presence of

N_1 in message (2) assures A that the correspondent is B.

3. A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_S and sends $M = E(PU_b, E(PR_a, K_S))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

UNIT - III

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

MESSAGE AUTHENTICATION

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. It is intended against the attacks like content modification, sequence modification, timing modification and repudiation. For repudiation, concept of digital signatures is used to counter it. There are three classes by which different types of functions that may be used to produce an authenticator. They are:

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

MESSAGE ENCRYPTION:

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

- Sometimes one needs to avoid encryption of full messages due to legal requirements.
- Encryption and authentication may be separated in the system architecture.
- The different ways in which message encryption can provide authentication, confidentiality in both symmetric and asymmetric encryption techniques is explained with the table below:

confidentiality and authentication implications of message authentication

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

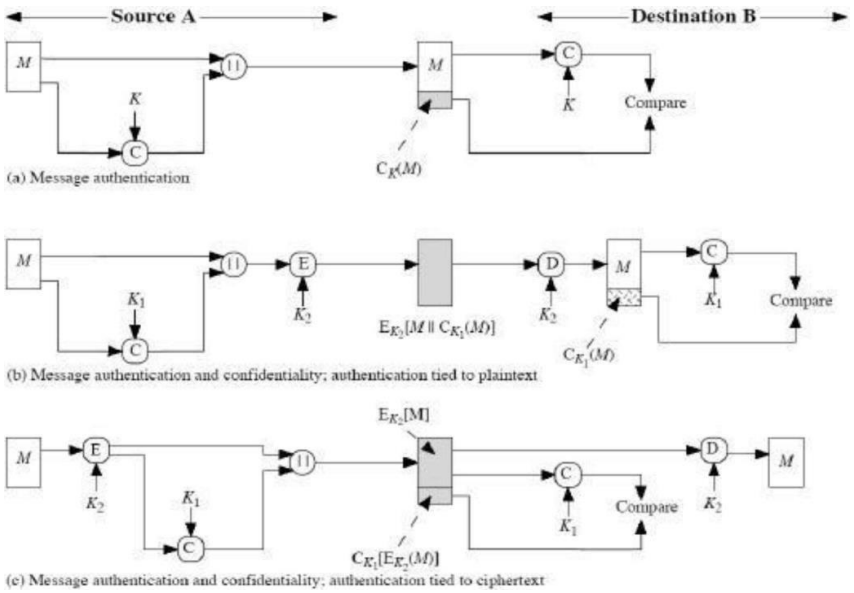
Message Authentication Code (MAC)

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as cryptographic checksum or MAC, which is appended to the message. This technique assumes that both the communicating parties say A and B share a common secret key K.

When A has a message to send to B, it calculates MAC as a function C of key and message given as: **MAC=Ck(M)** The message and the MAC are transmitted to the intended recipient, who upon receiving performs the same calculation on the received message, using the same secret key to generate a new MAC. The received MAC is compared to the calculated MAC and only if they match, then:

1. The receiver is assured that the message has not been altered: Any alternations been done the MAC's don't match.
2. The receiver is assured that the message is from the alleged sender: No one except the sender has the secret key and could prepare a message with a proper MAC.
3. If the message includes a sequence number, then receiver is assured of proper sequence as an attacker cannot successfully alter the sequence number.

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

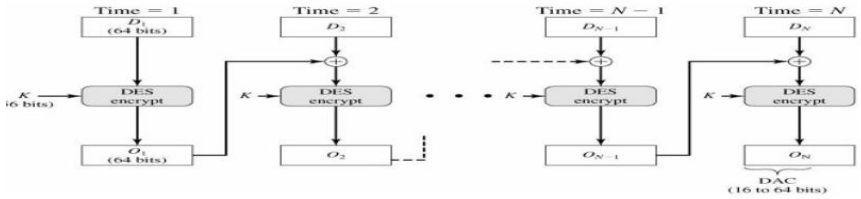


There are three different situations where use of a MAC is desirable:

1. If a message is broadcast to several destinations in a network (such as a military control center), then it is cheaper and more reliable to have just one node responsible to evaluate the authenticity –message will be sent in plain with an attached authenticator.
2. If one side has a heavy load, it cannot afford to decrypt all messages –it will just check the authenticity of some randomly selected messages.
3. Authentication of computer programs in plaintext is very attractive service as they need not be decrypted every time wasting of processor resources. Integrity of the program can always be checked by MAC

Message Authentication Code Based on DES

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



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

$$
\begin{aligned}
O_1 &= E(K, D_1) \\
O_2 &= E(K, [D_2 \oplus O_1]) \\
O_3 &= (K, [D_3 \oplus O_2]) \\
&\bullet \\
&\bullet \\
&\bullet \\
O_N &= E(K, [D_N \oplus O_{N1}])
\end{aligned}
$$

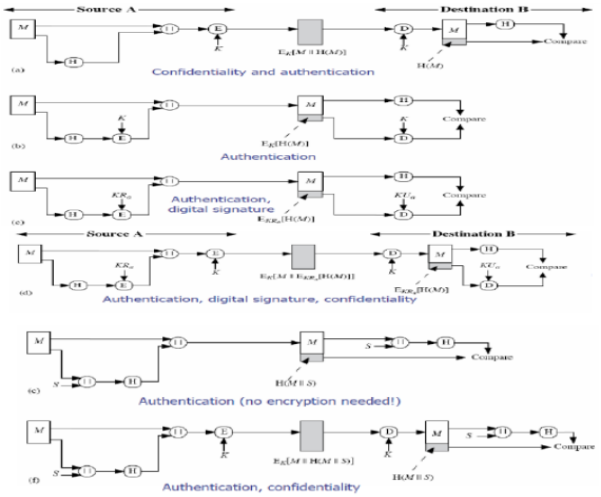
The DAC consists of either the entire block ON or the leftmost M bits of the block, with $16 \leq M \leq 64$ Use of MAC needs a shared secret key between the communicating parties and also MAC does not provide digital signature. The following table summarizes the confidentiality and authentication implications of the approaches shown above.

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

Hash function

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

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

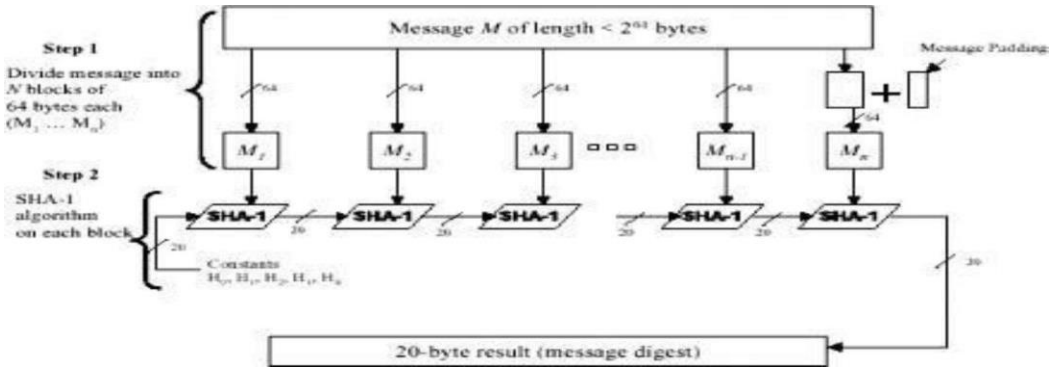


Encryption software is quite slow and may be covered by patents. Also encryption hardware costs are not negligible and the algorithms are subject to U.S export control.

- A fixed-length hash value h is generated by a function H that takes as input a message of arbitrary length**: h=H(M).**sends M and H(M)
- authenticates the message by computing H(M) and checking the match

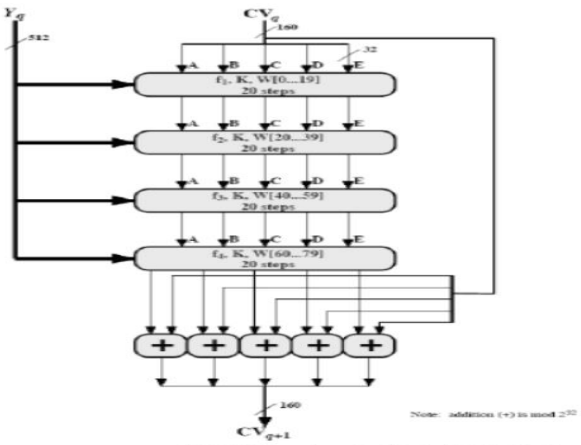
Secure Hash Algorithm

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



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

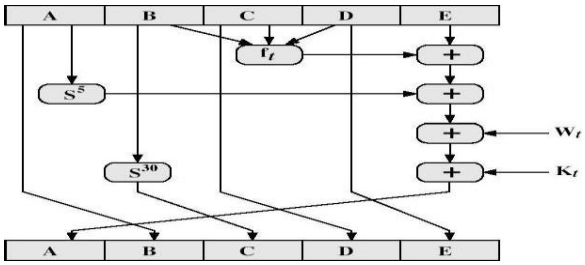
1. Append Padding Bits: The message is padded so that length is congruent to 448 modulo 512; padding always added –one bit 1 followed by the necessary number of 0 bits.
2. Append Length: a block of 64 bits containing the length of the original message is added
3. Initialize MD buffer: A 160-bit buffer is used to hold intermediate and final results on the hash function. This is formed by 32-bit registers A,B,C,D,E. Initial values: A=0x67452301, B=0xEFCDAB89, C=0x98BADCFE, D=0x10325476, E=C3D2E1F0. Stores in big-endian format i.e. the most significant bit in low address.
4. Process message in blocks 512-bit (16-word) blocks: The processing of a single 512-bit block is shown above. It consists of four rounds of processing of 20 steps each. These four rounds have similar structure, but uses a different primitive logical function, which we refer to as f1, f2, f3 and f4. Each round takes as input the current 512-bit block being processed and the 160-bit buffer value ABCDE and updates the contents of the buffer. Each round also makes use of four distinct additive constants Kt. The output of the fourth round i.e. eightieth step is added to the input to the first round to produce $CV_{q\,1}$.
5. Output: After all L 512-bit blocks have been processed, the output from the Lth stage is the 160-bit message digest.



Where, IV = initial value of ABCDE buffer
ABCDEq = output of last round of processing of qth message block
L = number of blocks in the message
SUM32 = Addition modulo 2^{32}
MD = final message digest value.

SHA-1 Compression Function:

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



SHA shares much in common with MD4/5, but with 20 instead of 16 steps in each of the 4 rounds. Note the 4 constants are based on sqrt(2,3,5,10). Note also that instead of just splitting the input block into 32-bit words and using them directly, SHA-1 shuffles and mixes

Them using roates & XOR's to form a more complex input, and greatly increases the difficulty of finding collisions. A sequence of logical functions f0, f1,..., f79 is used in the SHA-1. Each f**t**, 0<=t<=79, operates on three 32-bit words B, C, D and produces a 32-bit word as output. ft(B,C,D) is defined as follows: for words B, C, D, **ft(B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19) ft(B,C,D) = B XOR C XOR D (20 <= t <= 39) ft(B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59) ft(B,C,D) = B XOR C XOR D (60 <= t <= 79).**

WHIRLPOOL HASH FUNCTION

- ✓ Created by Vincent Rijmen and Paulo S. L. M. Barreto
  Hashes messages of plaintext length 2^256
- ✓ Result is a 512 bit message
- ✓ Three versions have been released WHIRLPOOL-0 WHIRLPOOL-T WHIRLPOOL
  - designed specifically for hash function use
    with security and efficiency of AES

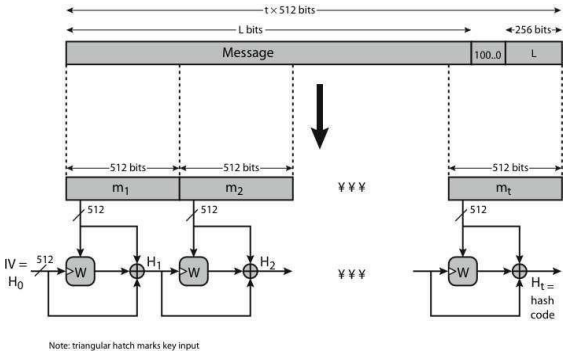
- but with 512-bit block size and hence hash similar structure & functions as AES but
  - ❖ input is mapped row wise has 10 rounds
  - ❖ a different primitive polynomial for GF(2^8) uses different S-box design & values

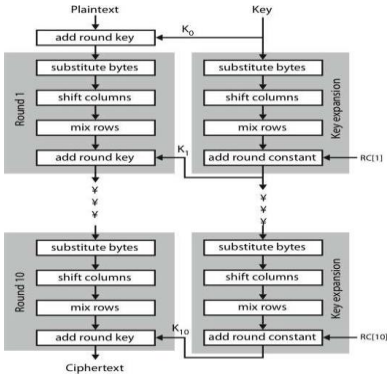
✓ "W" is a 512-bit block cipher

✓ "m" is the plaintext, split into 512 bit blocks

✓ "H" is the blocks formed from the hashes

WHIRLPOOL OVERVIEW





- The block cipher W is the core element of the Whirlpool hash function It is comprised of 4 steps.
    - ✓ Add Round Key
    - ✓ Shift Columns
    - ✓ Mix Rows
    - ✓ Substitute
    - ✓ bytes

During the add round key step, the message is XOR' key
If this is the first message block being run through, the key is a block of allzeros If this is any block except the first, the key is the digest of the previous block **Shift Columns**
Starting from left to right, each column gets rotated vertically a number of bytes equal to which number column it is, from top to bottom
Ex:

Mix Rows

[0,0][0,1][0,2] [0,0][2,1][1,2]
[1,0][1,1][1,2]----------> [1,0][0,1][2,2]
[2,0][2,1][2,2] [2,0][1,1][0,2]

Each row gets shifted horizontally by the number of row it is. Similar to the shift column function, but rotated left to right Ex:

[0,0][0,1][0,2] [0,0][0,1][0,2]
[1,0][1,1][1,2]----------> [1,2][1,0][1,2]
[2,0][2,1][2,2] [2,1][2,2][0,2]

Substitute bytes

Each byte in the message is passed through a set of s-boxes The output of this is then set to be the key for the next round

HMAC

Interest in developing a MAC, derived from a cryptographic hash code has been increasing mainly because hash functions are generally faster and are also not limited by export restrictions unlike block ciphers. Additional reason also would be that the library code for cryptographic hash functions is widely available. The original proposal is for incorporation of a secret key into an existing hash algorithm and the approach that received most support is HMAC. HMAC is specified as Internet standard RFC2104. It makes use of the hash function on the given message. Any of MD5, SHA-1, RIPEMD-160 can be used.

HMAC Design Objectives

1. To use, without modifications, available hash functions
2. To allow for easy replaceability of the embedded hash function
3. To preserve the original performance of the hash function
4. To use and handle keys in a simple way
5. To have a well understood cryptographic analysis of the strength of the MAC based on reasonable assumptions on the embedded hash function

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

Steps involved in HMAC algorithm:

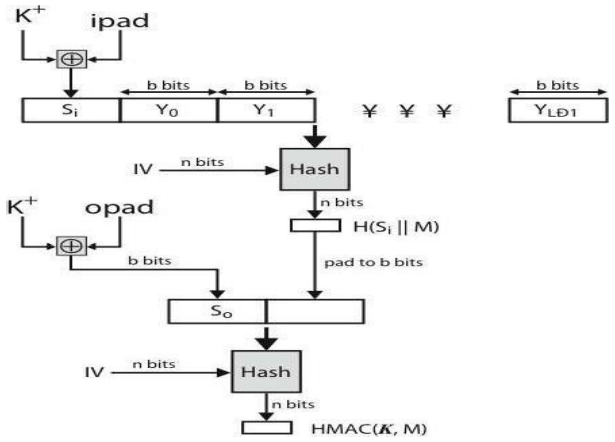
1. Append zeroes to the left end of K to create a b-bit string K (ex: If K is of length 160-bits and b = 512, then K will be appended with 44 zero bytes).
2. XOR(bitwise exclusive-OR) K with ipad to produce the b-bit block Si.
3. Append M to Si.
4. Now apply H to the stream generated in step-3
5. XOR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

- **Define the following terms**

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

- **Then HMAC can be expressed as**

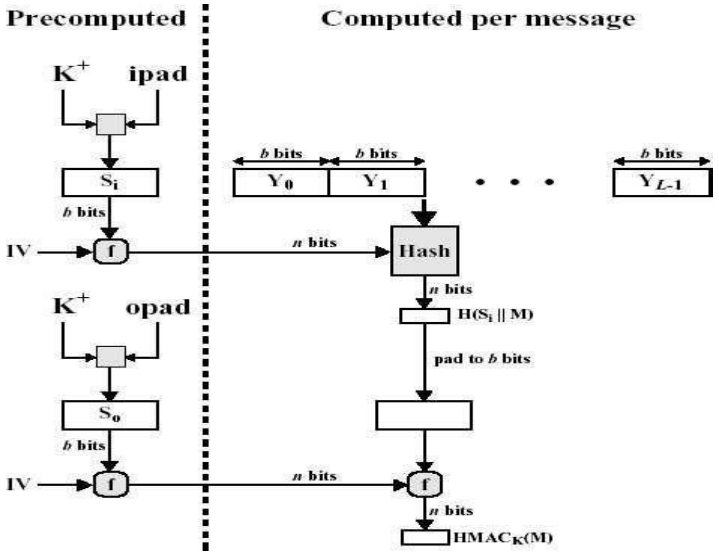
$$HMAC_K = H[\ (K^+ \oplus opad) \ ||\ H[K^+ \oplus ipad) \ ||\ M]\ ]$$



HMAC Structure

- The XOR with ipad results in flipping one-half of the bits of K. Similarly, XOR with opad results in flipping one-half of the bits of K, but different set of bits. By passing Si and S0 through the compression function of the hash algorithm, we have pseudo randomly generated two keys from K.
- HMAC should execute in approximately the same time as the embedded hash function for long messages. HMAC adds three executions of the hash compression function (for S0, Si, and the block produced from the inner hash)
- A more efficient implementation is possible. Two quantities are precomputed.

where f is the compression function for the hash function which takes as arguments a chaining variable of n bits and a block of b-bits and produces a chaining variable of n bits.



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

Digital Signature

The most important development from the work on public-key cryptography is the digital signature. Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. A digital signature is analogous to the handwritten signature, and provides a set of security capabilities that would be difficult to implement in any other way. It must have the following properties:

• It must verify the author and the date and time of the signature
• It must to authenticate the contents at the time of the signature
• It must be verifiable by third parties, to resolve disputes

Thus, the digital signature function includes the authentication function. A variety of approaches has been proposed for the digital signature function. These approaches fall into two categories: direct and arbitrated.

Direct Digital Signature

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

Arbitrated Digital Signature

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

Using Conventional encryption

$X \rightarrow A : M \,||\, E(K_{xa}, [ID_x \,||\, H(M)])$

$A \rightarrow Y : E(K_{ay}, [ID_x \,||\, M \,||\, E(K_{xa}, [ID_x \,||\, H(M))]) \,||\, T])$

- It is assumed that the sender X and the arbiter A share a secret key K_{xa} and that A and Y share secret key K_{ay}. X constructs a message M and computes its hash value H(m) . Then X transmits the message plus a signature to A. the signature consists of an identifier ID_x of X plus the hash value, all encrypted using K_{xa}.
- A decrypts the signature and checks the hash value to validate the message. Then A transmits a message to Y, encrypted with K_{ay}. The message includes ID_x, the original message from X, the signature, and a timestamp.
- Arbiter sees message Problem : the arbiter could form an alliance with sender to deny a signed message, or with the receiver to forge the sender's signature.

Using Public Key Encryption

$X \rightarrow A : ID_x \,||\, E(PR_x, [ID_x \,||\, E(PU_y, E(PR_x, M))])$

$A \rightarrow Y : E(PR_a, [ID_x \,||\, E(PU_y, E(PR_x, M)) \,||\, T])$

- X double encrypts a message M first with X's private key,PR_x, and then with Y's public key, PU_y. This is a signed, secret version of the message. This signed message, together with X's identifier , is encrypted again with PR_x and, together with IDx, is sent to A. The inner, double encrypted message is secure from the arbiter (and everyone else except Y)
- A can decrypt the outer encryption to assure that the message must have come from X (because only X has PR_x). Then A transmits a message to Y, encrypted with PR_a. The message includes ID_x, the double encrypted message, and a timestamp.
- Arbiter does not see message

AUTHENTICATION APPLICATIONS:

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

- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

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

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

Reliable: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service

means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

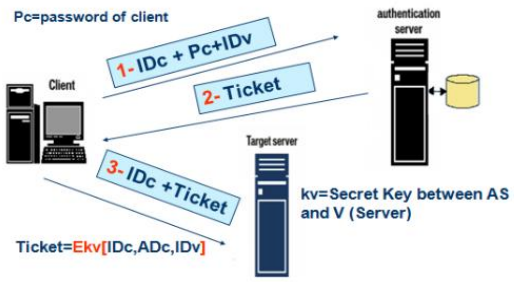
Transparent: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

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

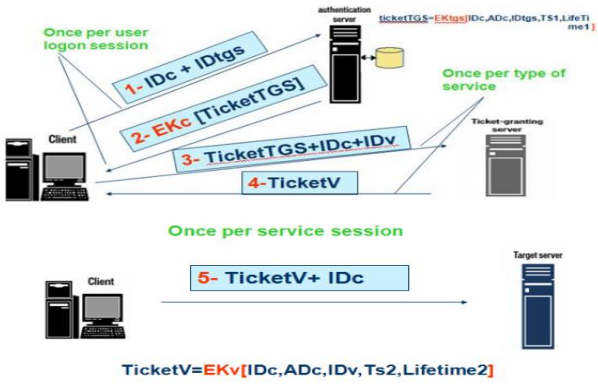
Two versions of Kerberos are in common use: Version 4 is most widely used version. Version 5 corrects some of the security deficiencies of Version 4. Version 5 has been issued as a draft Internet Standard (RFC 1510).

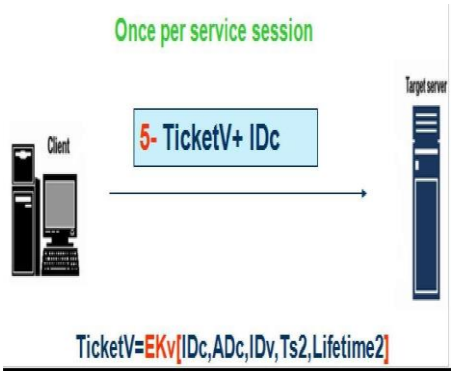
Kerberos dialogue:

Version 4: 1.) Simple



More Secure Dialogue





Kerberos Version 4 Authentication Dialogue

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

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

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

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

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

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

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

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

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

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

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

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

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

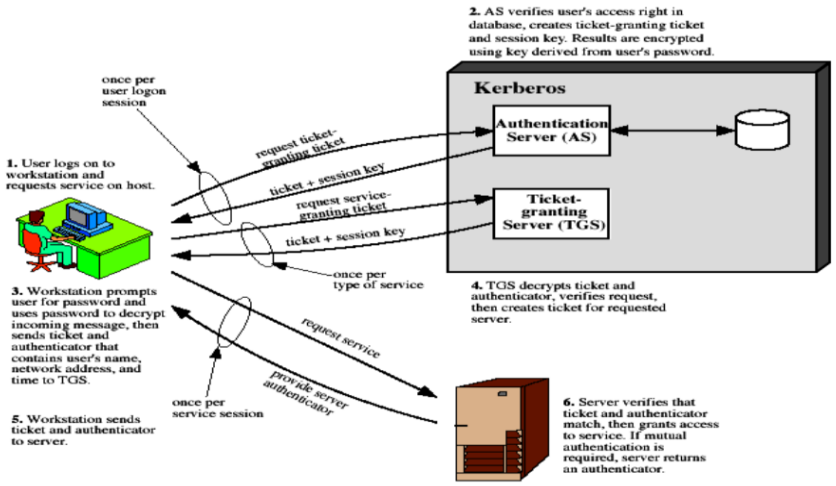
$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Client/Server Authentication Exchange to obtain service

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

- Message (1) includes a timestamp; so that the AS knows that the message is timely.
- Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the TGS and to learn its expiration time. Note that the ticket does not prove anyone's identity but is a way to distribute keys securely. It is the authenticator that proves the client's identity.
- Because the authenticator can be used only once and has a short lifetime, the threat of an opponent stealing both the ticket and the authenticator for presentation later is countered. C then sends the TGS a message that includes the ticket plus the ID of the requested service (message 3).
- The reply from the TGS, in message (4), follows the form of message (2). C now has a reusable service-granting ticket for V.
- When C presents this ticket, as shown in message (5), it also sends an authenticator.
- The server can decrypt the ticket, recover the session key, and decrypt the authenticator. If mutual authentication is required, the server can reply as shown in message (6).

Overview of Kerberos



Kerberos version 5

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

- Realm: Indicates realm of the user
- Options
- Times
  - From: the desired start time for the ticket
  - Till: the requested expiration time
  - Rtime: requested renew-till time
- Nonce: A random value to assure the response is fresh

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(c) Client/Server Authentication Exchange to obtain service

- Message (1) is a client request for a ticket-granting ticket.

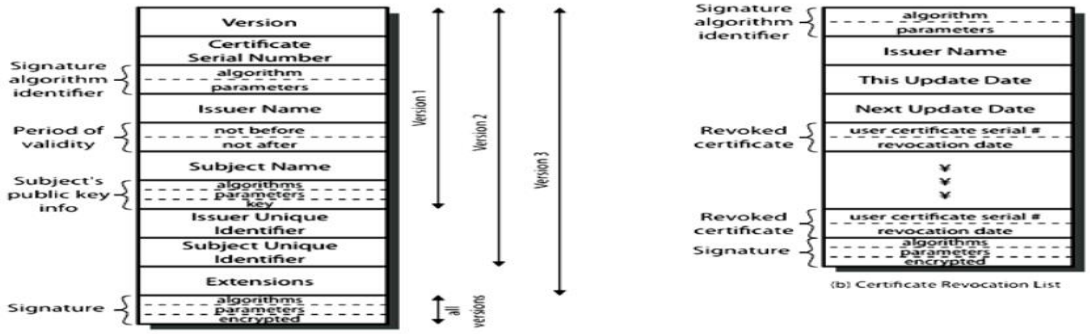
- Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password. This block includes the session key to be used between the client and the TGS.
- Now compare the **ticket-granting service** exchange for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

Advantages of Kerberos:

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

X.509 Authentication Service

- ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of 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 is based on the use of public-key cryptography and digital signatures.
- The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.



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

- version 1, 2, or 3
- serial number (unique within CA) identifying certificate
- signature algorithm identifier
- issuer X.500 name (CA)
- period of validity (from - to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2)
- subject unique identifier (v2)
- extension fields (v3)

- signature (of hash of all fields in certificate)

The standard uses the following notation to define a certificate:

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

Where,

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

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

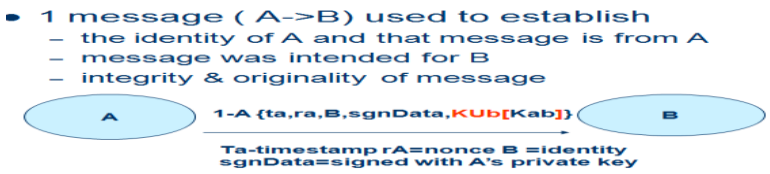
User certificates generated by a CA have the following characteristics:

- Any user with CA's public key can verify the user public key that was certified
- No party other than the CA can modify the certificate without being detected
- because they cannot be forged, certificates can be placed in a public directory

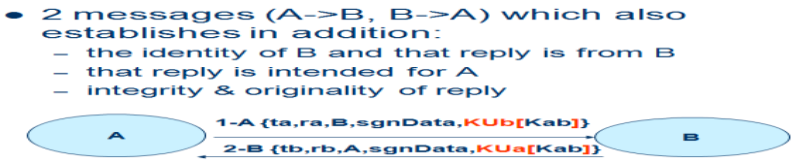
Authentication Procedures

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications. All these procedures make use of public-key signatures. It is assumed that the two parties know each other's public key, either by obtaining each other's certificates from the directory or because the certificate is included in the initial message from each side.

1. One-Way Authentication: One way authentication involves a single transfer of information from one user (A) to another (B), and establishes the details shown above. Note that only the identity of the initiating entity is verified in this process, not that of the responding entity. At a minimum, the message includes a timestamp, a nonce, and the identity of B and is signed with A's private key. The message may also include information to be conveyed, such as a session key for B.



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



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

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.

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 recognize entities by an Internet e-mail address, a URL, or some other Internet-related identification.

3. There is a need to indicate security policy information. 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 life cycle 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. X.509 version 3 includes a number of optional extensions that may be added to the version 2 format. Each extension consists of an extension identifier, a criticality indicator, and an extension value. The criticality indicator indicates whether an extension can be safely ignored or not.

BIOMETRIC AUTHENTICATION

Biometric authentication is a type of system that relies on the unique biological characteristics of individuals to verify identity for secure access to electronic systems.

Biometric verification is considered a subset of biometric authentication. The biometric technologies involved are based on the ways in which individuals can be uniquely identified through one or more distinguishing biological traits, such as fingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves, keystroke dynamics, DNA and signatures. Biometric authentication is the application of that proof of identity as part of a process validating a user for access to a system. Biometric technologies are used to secure a wide range of electronic communications, including enterprise security, online commerce and banking -- even just logging in to a computer or smartphone.

Biometric authentication systems compare the current biometric data capture to stored, confirmed authentic data in a database. If both samples of the biometric data match,
authentication is confirmed and access is granted. The process is sometimes part of amultifactor authentication system. For example, a smartphone user might log on with his personal identification number (PIN) and then provide an iris scan to complete the authentication process.

Types of biometric authentication technologies:

Retina scan: Iris recognition is used to identify individuals based on unique patterns within the ring-shaped region surrounding the pupil of the eye.

Fingerscanning, the digital version of the ink-and-paper fingerprinting process, works with details in the pattern of raised areas and branches in a human finger image.

Finger vein ID is based on the unique vascular pattern in an individual's finger.

Facial recognition systems work with numeric codes called faceprints, which identify 80 nodal points on a human face.

Voice identification systems rely on characteristics created by the shape of the speaker's mouth and throat, rather than more variable conditions.

Once seen mostly in spy movies (where it might be used to protect access to a top- secret military lab, for example), biometric authentication is becoming relatively commonplace. In addition to the security provided by hard-to-fake individual biological traits, the acceptance of biometric verification has also been driven by

The history of biometric verification:

- The oldest known use of biometric verification is fingerprinting. Thumbprints made on clay seals were used as a means of unique identification as far back as ancient China.
- Modern biometric verification has become almost instantaneous, and is increasingly accurate with the advent of computerized databases and the digitization of analog data.
- The market for biometrics products is still too fractured to name specific top providers. The physical characteristics of the biometrics products available today vary from the mundane, such as fingerprinting,

to the esoteric, like typing speeds and electrophysiological signals

Until recently, biometrics was typically used at a physical security level-protecting facilities at military bases or impenetrable bank vaults, for example. But, because single- factor authentication methods are easy to break, companies have started looking to two-factor solutions, like biometrics.

However, the following five fundamental barriers may limit the growth of biometric authentication:

1. Biometrics can be complicated and costly to deploy. All biometric deployments require installation of their own hardware and application servers.
2. The market is still fractured. Should you buy a fingerprint reader, a voice recognition system or an iris scanner? Since each product differs greatly in its approach and installation, it is difficult to compare them during a typical company bid process.
3. Biometric data is like any other data. It sits on servers, which are bait forhackers if not properly hardened and secured. Therefore, when reviewing any biometric product, make sure it transmits data securely, meaningencrypted, from the biometric reader back to the authenticating server. And, make sure the authenticating server has been hardened, patched and protected.
4. Biometric readers are prone to errors. Fingerprints can smudge, faces and voices can be changed and all of them can be misread, blocking a legitimate user, or permitting access to an unauthorized or malicious user.
5. Difficulties with user acceptance. Properly trained employees may be willing to use biometrics devices, but customers, like those logging on to your Web site, may be more reluctant to useor worse, forced to purchase a device that's difficult to use or makes doing business, such as banking, on your site, a hassle instead of a convenience. And both your employees and customers may be squeamish about exposing their eyes to devices like iris scanners, even if they appear harmless.

Despite these issues, biometrics is slowly gaining acceptance for two-factor authentication purposes. The products are getting better, lighter and easier to use. Error rates are going down, and fingerprint readers installed on tokens and laptops are getting smaller and less intrusive. And, like the rest of the security product industry, vendors will eventually merge and consolidate, uniting a fractured market, which will make it easier to choose a product that suits your business needs

UNIT-IV

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

Pretty Good Privacy:

In virtually all distributed environments, electronic mail is the most heavily used network-based application. But current email services are roughly like "postcards", anyone who wants could pick it up and have a look as it's in transit or sitting in the recipients mailbox. PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. With the explosively growing reliance on electronic mail for every conceivable purpose, there grows a demand for authentication and confidentiality services.

The Pretty Good Privacy (PGP) secure email program, is a remarkable phenomenon, has grown explosively and is now widely used. Largely the effort of a single person, Phil Zimmermann, who selected the best available crypto algorithms to use & integrated them into a single program, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. It is independent of government organizations and runs on a wide range of systems, in both free & commercial versions.

There are five important services in PGP

- Authentication (Sign/Verify)
- Confidentiality (Encryption/Decryption)
- Compression
- Email compatibility
- Segmentation and Reassembly

The last three are transparent to the user

PGP Notations:

Ks	**=session key used in symmetric encryption scheme**
PRa	**=private key of user A, used in public-key encryption scheme**
PUa	**=public key of user A, used in public-key encryption scheme**
EP	**= public-key encryption**
DP	**= public-key decryption**
EC	**= symmetric encryption**
DC	**= symmetric decryption**
H	**= hash function**
||	**= concatenation**
Z	**=compression using ZIP algorithm**
R64	**= conversion to radix 64 ASCII format**

PGP Operation –Compression

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

The signature is generated before compression for two reasons:

1. So that one can store only the uncompressed message together with signature for later verification
2. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm as the PGP compression algorithm is not deterministic
3. Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

PGP Operation –Email Compatibility

When PGP is used, at least part of the block to be transmitted is encrypted, and thus consists of a stream of arbitrary 8-bit octets. However many electronic mail systems only permit the use of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. It uses radix-64 conversion, in which each group of three octets of binary data

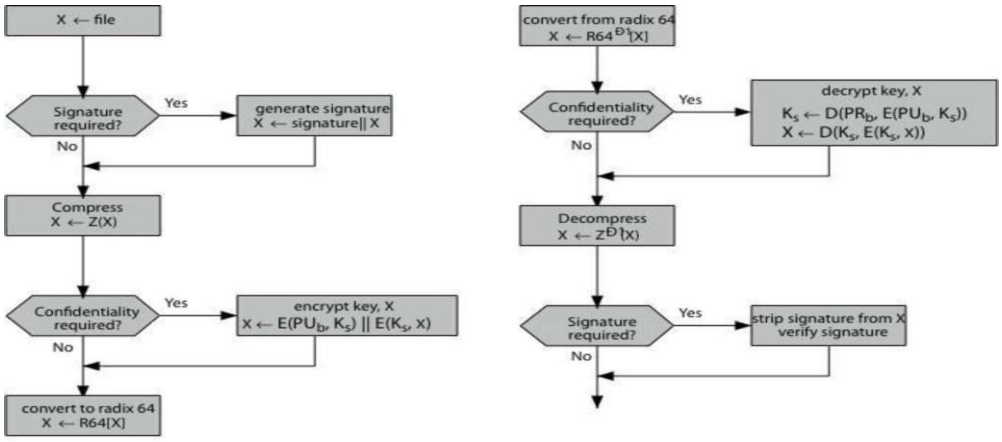
is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors. The use of radix 64 expands a message by 33%, but still an overall compression of about one- third can be achieved.

PGP Operation - Segmentation/Reassembly

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

PGP Operations Summary:

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



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

PGP Message Format

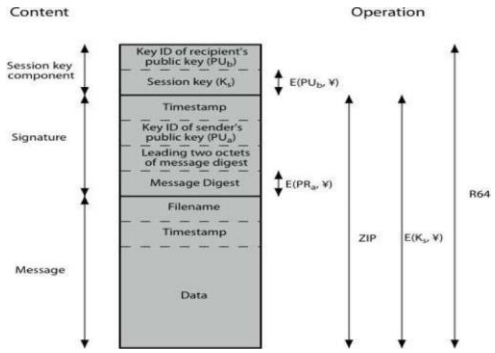
A message consists of three components: the message component, a signature (optional), and a session key component (optional). The *message component* includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation. The *signature component*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

- *Timestamp*: The time at which the signature was made.
- *Message digest*: The 160-bit SHA-1 digest, encrypted with the sender's private signature key.

Leading two octets of message digest: To enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication, by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.

Key ID of sender's public key: Identifies the public key that should be used to decrypt the

message digest and, hence, identifies the private key that was used to encrypt the message digest

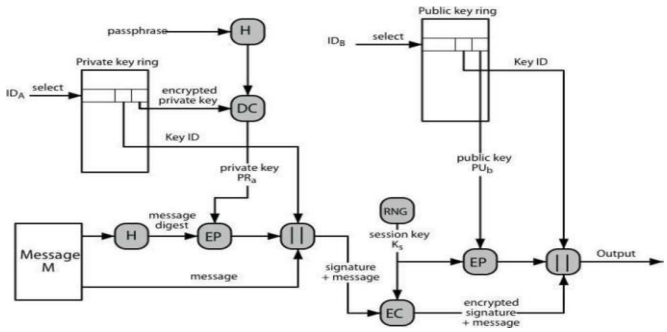


The *session key component* includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key. The entire block is usually encoded with radix-64 encoding.

PGP Message Transmission and Reception

Message transmission

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



The sending PGP entity performs the following steps:

Signing the message

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

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

c. The signature component of the message is constructed

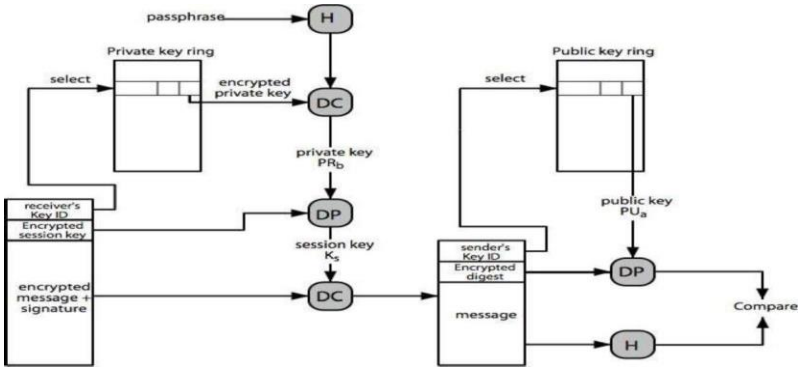
Encrypting the message

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

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

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

Message Reception



The receiving PGP entity performs the following steps:

Decrypting the message

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

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

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

Authenticating the message

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

b. PGP recovers the transmitted message digest.

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

S/MIME

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

RFC 822

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

Multipurpose Internet Mail Extensions

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

Problems with RFC 822 and SMTP

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

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

• Some servers reject mail messages over a certain size

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

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

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

The MIME specification includes the following elements:

1. Five new message header fields are defined, which provide information about the 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 protect the content from alteration by the mail system.

MIME - New header fields

The five header fields defined in MIME are as follows:

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

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

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

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

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

MIME Content Types

The bulk of the MIME specification is concerned with the definition of a variety of content types. There are seven different major types of content and a total of 15 subtypes. In general, a content type declares the general type of data, and the subtype specifies a particular format for that type of data.

For the text type of body, the primary subtype is plain text, which is simply a string of ASCII characters or ISO 8859 characters. The enriched subtype allows greater formatting flexibility.

The multipart type indicates that the body contains multiple, independent parts. The Content-Type header field includes a parameter called boundary that defines the delimiter between body parts. This boundary should not appear in any parts of the message. Each boundary starts on a new line and consists of two hyphens followed by the boundary value. The final boundary, which indicates the end of the last part, also has a suffix of two hyphens. Within each part, there may be an optional ordinary MIME header. There are four subtypes of the multipart type, all of which have the same overall syntax.

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.

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.

The message type provides a number of important capabilities in MIME. The message/rfc822 subtype indicates that the body is an entire message, including header and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 822 message, but also any MIME message. The message/partial subtype enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this subtype, three parameters are specified in the Content-Type: Message/Partial field: an id common to all fragments of the same message, a sequence number unique to each fragment, and the total number of fragments. The message/external-body subtype indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. The application type refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-based application.

MIME Transfer Encodings

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

MIME Transfer Encodings

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

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

encoding schemes defined are quoted-printable and base64. Two schemes are defined to provide a choice between a transfer technique that is essentially human readable and one that is safe for all types of data in a way that is reasonably compact.

The quoted-printable transfer encoding is useful when the data consists largely of octets that correspond to printable ASCII characters. In essence, it represents non safe 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.

MIME canonical form

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

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

S/MIME Functionality: S/MIME provides the following functions:

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

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

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

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

IP Security Overview

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

Need for IPSec

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

IPsec protection involves five main components:

- **Security protocols –** The IP datagram protection mechanisms. The authentication header (AH) signs IP packets and ensures integrity. The content of the datagram is not encrypted, but the receiver is assured that the packet contents have not been altered. The receiver is also assured that the packets were sent by the sender. The encapsulating security payload (ESP) encrypts IP data, thus obscuring the content during packet transmission. ESP also can ensure data integrity through an authentication algorithm option.

- **Security associations database (SADB) –** The database that associates a security protocol with an IP destination address and an indexing number. The indexing number is called the security parameter index (SPI). These three elements (the security protocol, the destination address, and the SPI) uniquely identify a legitimate IPsec packet. The database ensures that a protected packet that arrives to the packet destination is recognized by the receiver. The receiver also uses information from the database to decrypt the communication, verify that the packets are unchanged, reassemble the packets, and deliver the packets to their ultimate destination.
- **Key management –** The generation and distribution of keys for the cryptographic algorithms and for the SPI.
- **Security mechanisms –** The authentication and encryption algorithms that protect the data in the IP datagrams.
- **Security policy database (SPD) –** The database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine how the packets should be processed. A packet can be discarded. A packet can be passed in the clear. Or, a packet can be protected with IPsec. For outbound packets, the SPD and the SADB determine what level of protection to apply. For inbound packets, the SPD helps to determine if the level of protection on the packet is acceptable. If the packet is protected by IPsec, the SPD is consulted after the packet has been decrypted and has been verified.

IPsec applies the security mechanisms to IP datagrams that travel to the IP destination address. The receiver uses information in its SADB to verify that the arriving packets are legitimate and to decrypt them. Applications can invoke IPsec to apply security mechanisms to IP datagrams on a per-socket level as well.

IPsec Terminology

The IPsec RFCs define a number of terms that are useful to recognize when implementing IPsec on your systems. The following table lists IPsec terms, provides their commonly used acronyms, and defines each term.

IPsec Term	Acronym	Definition
Security association	SA	A unique connection between two nodes on a network. The connection is defined by a triplet: a security protocol, a security parameter index, and an IP destination. The IP destination can be an IP address or a socket.
Security associations database	SADB	Database that contains all active security associations.
Security parameter index	SPI	The indexing value for a security association. An SPI is a 32-bit value that distinguishes among SAs that have the same IP destination and security protocol.
Security policy database	SPD	Database that determines if outbound packets and inbound packets have the specified level of protection.
Key exchange		The process of generating keys for asymmetric cryptographic algorithms. The two main methods are RSA protocols and the Diffie-Hellman protocol.
Diffie-Hellman protocol	DH	A key exchange protocol that involves key generation and key authentication. Often called **authenticated key exchange**.
RSA protocol	RSA	A key exchange protocol that involves key generation and key distribution. The protocol is named for its three creators, Rivest, Shamir, and Adleman.
Internet Security Association and Key Management Protocol	ISAKMP	The common framework for establishing the format of SA attributes, and for negotiating, modifying, and deleting SAs. ISAKMP is the IETF standard for handling IPsec SAs.

Applications of IPSec

IPSec provides the capability to secure communications across a LAN, across private and public wide area networks (WAN's), and across the Internet.

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

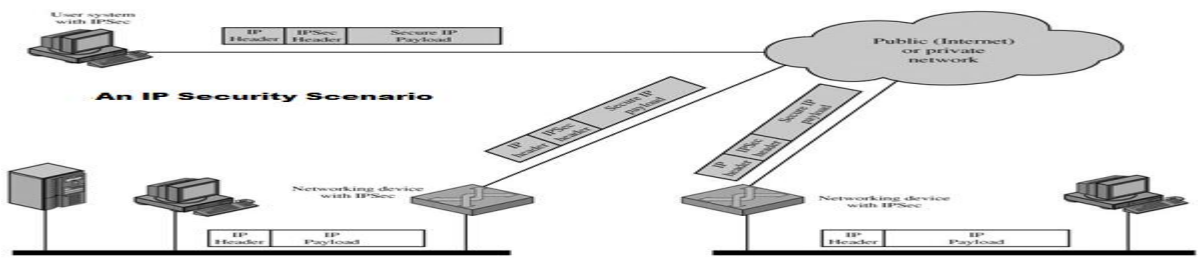
• **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for travelling employees and telecommuters.

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

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

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

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



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

Benefits of IPSec

The benefits of IPSec are listed below:

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

• IPSec in a firewall is resistant to bypass

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

• IPSec can be transparent to end users

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

Routing Applications

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

• router advertisements come from authorized routers

• neighbor advertisements come from authorized routers

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

• A routing update is not forged

IP Security Architecture

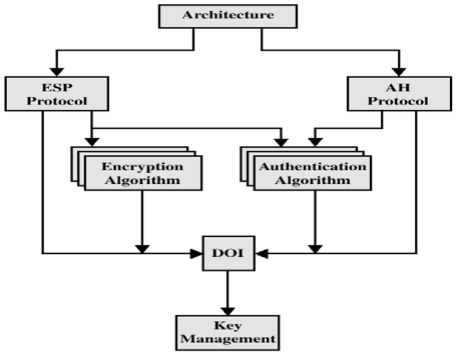
To understand IP Security architecture, we examine IPSec documents first and then move on to IPSec services and Security Associations.

IPSec Documents

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

• RFC 2401: An overview of a security architecture
• RFC 2402: Description of a packet authentication extension to IPv4 and IPv6
• RFC 2406: Description of a packet encryption extension to IPv4 and IPv6
• RFC 2408: Specification of key management capabilities

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



• **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology
• **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.

• **Authentication Header (AH):** Covers the packet format and general issues related to the use of AH for packet authentication.
• **Encryption Algorithm:** A set of documents that describe how various encryption algorithms are used for ESP.
• **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.
• **Key Management:** Documents that describe key management schemes.
• **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

IPSec Services

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

- **Connectionless Integrity:-** Data integrity service is provided by IPSec via AH which prevents the data from being altered during transmission.
- **Data Origin Authentication:-** This IPSec service prevents the occurrence of replay attacks, address spoofing etc., which can be fatal
- **Access Control:-** The cryptographic keys are distributed and the traffic flow is controlled in both AH and ESP protocols, which is done to accomplish access control over the data transmission.
- **Confidentiality:-** Confidentiality on the data packet is obtained by using an encryption technique in which all the data packets are transformed into ciphertext packets which are unreadable and difficult to understand.
- **Limited Traffic Flow Confidentiality:-** This facility or service provided by IPSec ensures that the confidentiality is maintained on the number of packets transferred or received. This can be done using padding in ESP.
- **Replay packets Rejection:-** The duplicate or replay packets are identified and discarded using the sequence number field in both AH and ESP.

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

Security Associations

Since IPSEC is designed to be able to use various security protocols, it uses Security Associations (SA) to specify the protocols to be used. SA is a database record which specifies security parameters controlling security operations. They are referenced by the sending host and established by the receiving host. An index parameter called the Security Parameters Index (SPI) is used. SAs are in one direction only and a second SA must be established for the transmission to be bi-directional. A security association is uniquely identified by three parameters:
• **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.
• **IP Destination Address:** Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.
• **Security Protocol Identifier:** This indicates whether the association is an AH or ESP security association.

SA Parameters

In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:
• **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or

ESP headers

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

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

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

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

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

• **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations).

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

Transport and Tunnel Modes

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

	Transport Mode SA	Tunnel Mode SA
AH	**Authenticates** IP payload and selected portions of IP header and IPv6 extension headers	**Authenticates** entire inner IP packet plus selected portions of outer IP header
ESP	**Encrypts** IP payload and any IPv6 extesion header	**Encrypts** inner IP packet
ESP with authentication	**Encrypts** IP payload and any IPv6 extesion header. **Authenticates** IP payload but no IP header	**Encrypts** inner IP packet. **Authenticates** inner IP packet.

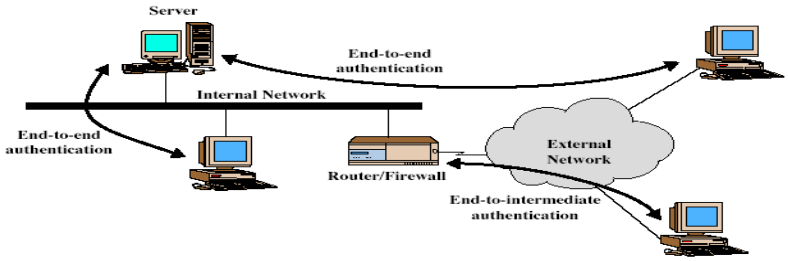
IP sec can be used (both AH packets and ESP packets) in two modes

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

1. Typically used in end-to-end communication
2. IP header not protected

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

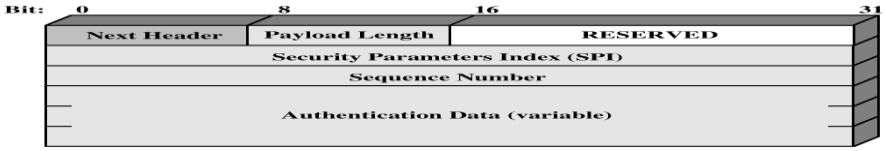
1. Typically used in firewall-to-firewall communication
2. Provides protection for the whole IP packet
3. No routers along the way will be able (and will not need) to check the content of the packets



End-to-End versus End-to-Intermediate Authentication

IPSec Authentication Header:

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



IPSec Authentication Header

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

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

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

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

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

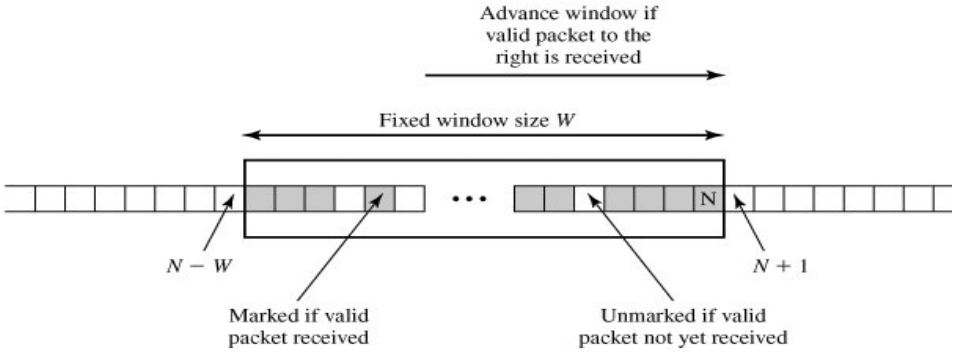
• **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.

Anti-Replay Service

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

When a new SA is established, the sender initializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. This value goes on increasing with respect to the number of packets being transmitted. The sequence number field in each packet represents the value of this counter. The maximum value of the sequence number field can go up to 2^{32}-1. If the limit of 2^{32}-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key.

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



Antireplay Mechanism

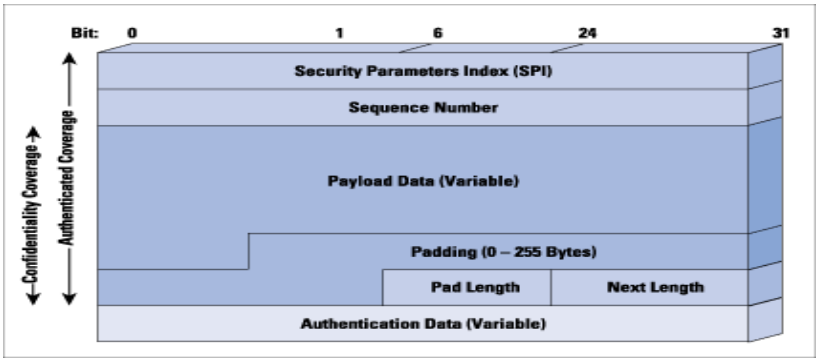
1. If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.
2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced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

Encapsulating Security Payload

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

ESP Format

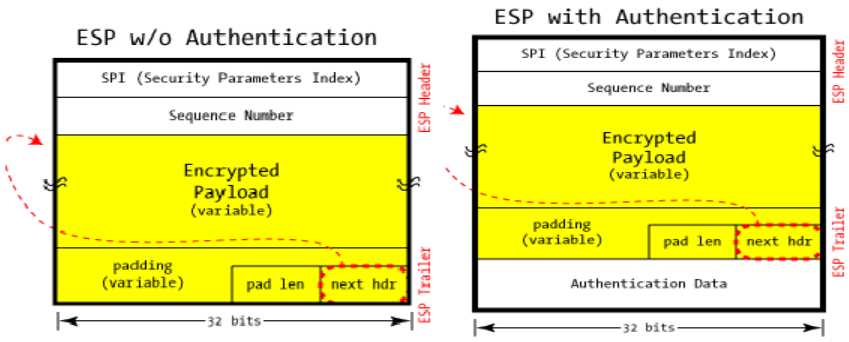
The following figure shows the format of an ESP packet. It contains the following fields:



IPSec ESP format

• **Security Parameters Index (32 bits):** Identifies a security association.
• **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
• **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
• **Padding (0-255 bytes):** This field is used to make the length of the plaintext to be a multiple of some desired number of bytes. It is also added to provide confidentiality.
• **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
• **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).
• **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

Adding encryption makes ESP a bit more complicated because the encapsulation surrounds the payload rather than precedes it as with AH: ESP includes header and trailer fields to support the encryption and optional authentication. It also provides Tunnel and Transport modes. The IPSec RFCs don't insist upon any particular encryption algorithms, but we find DES, triple-DES, AES, and Blowfish in common use to shield the payload from prying eyes. The algorithm used for a particular connection is specified by the Security Association and this SA includes not only the algorithm, but the key used. Unlike AH, which provides a small header before the payload, ESP surrounds the payload it's protecting. The Security Parameters Index and Sequence Number serve the same purpose as in AH, but we find padding, the next header, and the optional Authentication Data at the end, in the ESP Trailer.

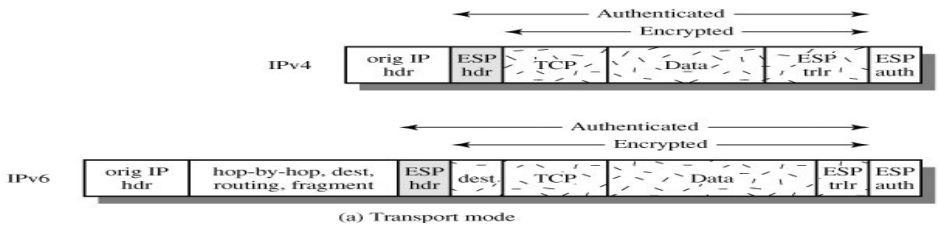


It's possible to use ESP without any actual encryption (to use a NULL algorithm), which nonetheless structures the packet the same way. This provides no confidentiality, and it only makes sense if combined with ESP authentication. Padding is provided to allow block-oriented encryption algorithms room for multiples of their block size, and the length of that padding is provided in the pad len field. The next hdr field gives the type (IP, TCP, UDP, etc.) of the payload in the usual way, though it can be thought of as pointing "backwards" into the

packet rather than forward as we've seen in AH. In addition to encryption, ESP can also optionally provide authentication, with the same HMAC as found in AH. Unlike AH, however, this authentication is only for the ESP header and encrypted payload: it does not cover the full IP packet.

Transport Mode ESP

Transport mode ESP is used to encrypt and optionally authenticate the data carried by IP (e.g., a TCP segment). For this mode using IPv4, the ESP header is inserted into the IP packet immediately prior to the transport-layer header (e.g., TCP, UDP, ICMP) and an ESP trailer (Padding, Pad Length, and Next Header fields) is placed after the IP packet; if authentication is selected, the ESP Authentication Data field is added after the ESP trailer. The entire transport-level segment plus the ESP trailer are encrypted. Authentication covers all of the ciphertext plus the ESP header. In the context of IPv6, ESP is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the ESP header appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the ESP header, depending on the semantics desired. For IPv6, encryption covers the entire transport-level segment plus the ESP trailer plus the destination options extension header if it occurs after the ESP header. Again, authentication covers the ciphertext plus the ESP header.



(a) Transport mode

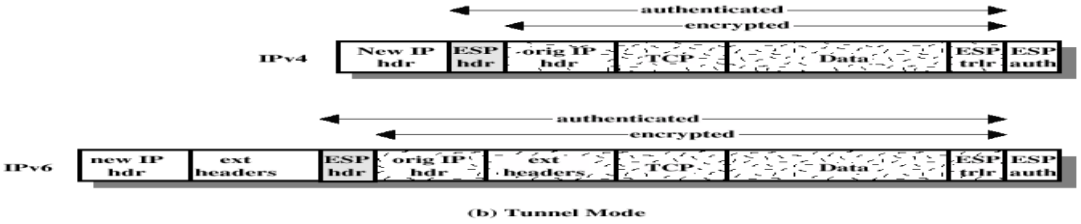
Transport mode operation may be summarized as follows:

1. At the source, the block of data consisting of the ESP trailer plus the entire transport-layer segment is encrypted and the plaintext of this block is replaced with its ciphertext to form the IP packet for transmission. Authentication is added if this option is selected.
2. The packet is then routed to the destination. Each intermediate router needs to examine and process the IP header plus any plaintext IP extension headers but does not need to examine the ciphertext.
   3. The destination node examines and processes the IP header plus any plaintext IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext transport-layer segment.

3. Transport mode operation provides confidentiality for any application that uses it, thus avoiding the need to implement confidentiality in every individual application. This mode of operation is also reasonably efficient, adding little to the total length of the IP packet. One drawback to this mode is that it is possible to do traffic analysis on the transmitted packets.

Tunnel Mode ESP

In case of tunnel mode ESP, ESP header and the ESP trailer are attached before and after the IP packet respectively, then the complete IP packet which includes IP header, Transport header and data field along with the ESP trailer is encrypted. Tunnel mode ESP is used to protect against the traffic flow analysis. But if ESP header precedes the IP header, the routers cannot identify and process this packet as the routing information and other parameters needed are present in the IP header of the packet. To overcome this problem,the complete structure which contains ESP header, encrypted text as well as authentication data are encapsulated in a new IP packet with a new IP header. This new IP header has enough routing information inorder to process the packet to the appropriate destination.



(b) Tunnel Mode

The transport mode is suitable for protecting connections between hosts that support the ESP feature and the tunnel mode is useful in a configuration that includes a firewall or other sort of security gateway that protects a trusted network from external networks. Consider a case in which an external host wishes to communicate with a host on an internal network protected by a firewall, and in which ESP is implemented in the external host and the firewalls. The following steps occur for transfer of a transport-layer segment from the external host to the internal host:

1. The source prepares an inner IP packet with a destination address of the target internal host. This packet is prefixed by an ESP header; then the packet and ESP trailer are encrypted and Authentication Data may be added. The resulting block is encapsulated with a new IP header (base header plus optional extensions such as routing and hop-by-hop options for IPv6) whose destination address is the firewall; this forms the outer IP packet.

2. The outer packet is routed to the destination firewall. Each intermediate router needs to examine and process the outer IP header plus any outer IP extension headers but does not need to examine the ciphertext.

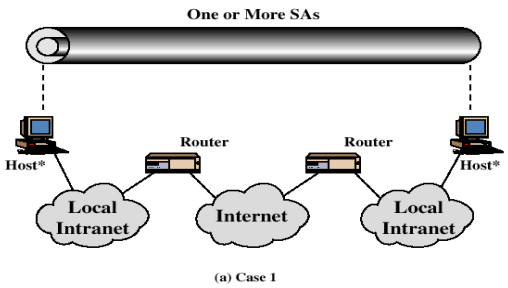
3. The destination firewall examines and processes the outer IP header plus any outer IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext inner IP packet. This packet is then transmitted in the internal network.

4. The inner packet is routed through zero or more routers in the internal network to the destination host.

Basic Combinations of Security Associations

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

Case 1 :-



(a) Case 1

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

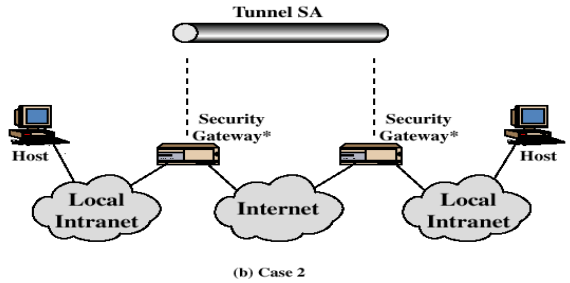
a) AH in transport mode

b) ESP in transport mode

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

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

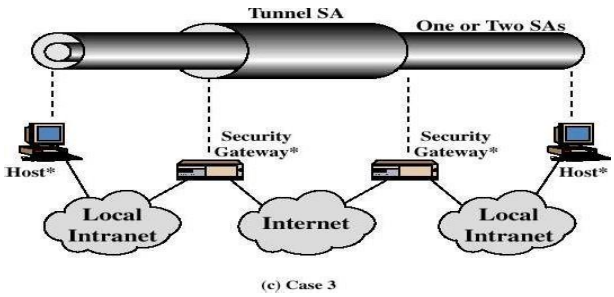
Case 2 :-



(b) Case 2

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

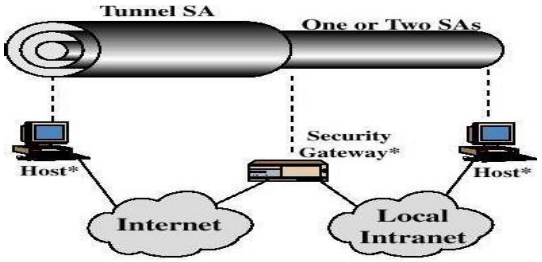
Case-3:-



(c) Case 3

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

Case:-4



(d) Case 4

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

Key Management

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

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

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

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

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

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

Oakley Key Determination Protocol

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

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

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

However, Diffie-Hellman has got some weaknesses:

• No identity information about the parties is provided.

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

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

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

Features of Oakley

The Oakley algorithm is characterized by five important features:

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

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

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

Aggressive Oakley Key Exchange

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

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

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

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

Notation:

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

Example of Aggressive Oakley Key Exchange

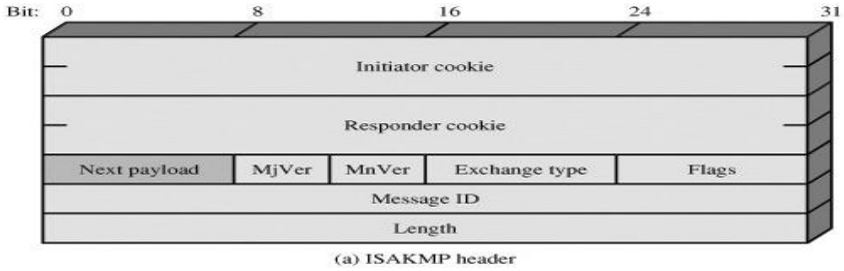
In the first step, the initiator (I) transmits a cookie, the group to be used, and I's public Diffie-Hellman key for this exchange. I also indicates the offered public-key encryption, hash, and authentication algorithms to be used in this exchange. Also included in this message are the identifiers of I and the responder (R) and I's nonce for this exchange. Finally, I appends a signature using I's private key that signs the two identifiers, the nonce, the group, the Diffie-Hellman public key, and the offered algorithms.

When R receives the message, R verifies the signature using I's public signing key. R acknowledges the message by echoing back I's cookie, identifier, and nonce, as well as the group. R also includes in the message a cookie, R's Diffie-Hellman public key, the selected algorithms (which must be among the offered algorithms), R's identifier, and R's nonce for this exchange. Finally, R appends a signature using R's private key that signs the two identifiers, the two nonces, the group, the two Diffie-Hellman public keys, and the selected algorithms.

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

ISKAMP

ISAKMP defines procedures and packet formats to establish, negotiate, modify, and delete security associations. As part of SA establishment, ISAKMP defines payloads for exchanging key generation and authentication data. **ISAKMP Header Format:** An ISAKMP message consists of an ISAKMP header followed by one or more payloads and must follow UDP transport layer protocol for its implementation. The header format of an ISAKMP header is shown below:



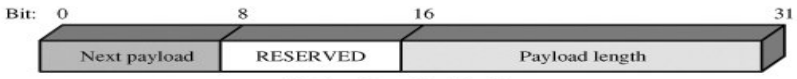
(a) ISAKMP header

- **Initiator cookie (64 bits):** Cookie of entity that initiated SA establishment, SA notification, or SA deletion.
- **Responder cookie (64 bits):** Cookie of responding entity; null in first message from initiator.

- **Next Payload(8 bits):** Indicates the type of the first payload in the message
    - **Major Version(4 bits):** Indicates major version of ISAKMP in use.
    - **Minor Version(4 bits):** Indicates minor version in use.
- **Exhange Types(8 bits):** Indicates the type of exchange. Can be informational, aggressive, authentication only, identity protection or base exchange (S).
- **Flags(8 bits):** Indicates specific options set for this ISAKMP exchange. Two bits so far defined: The Encryption bit is set if all payloads following the header are encrypted using the encryption algorithm for this SA. The Commit bit is used to ensure that encrypted material is not received prior to completion of SA establishment.
    - **Message ID(32 bits):** Unique ID for this message.
    - **Length(32 bits):** Length of total message (header plus all payloads) in octets

ISAKMP Payload Types

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



(b) Generic payload header

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

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

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

c. The **Transform payload** defines a security transform to be used to secure the communications channel for the designated protocol. The Transform # parameter serves to identify this particular payload so that the responder may use it to indicate acceptance of this transform. The Transform-ID and Attributes fields identify a specific transform (e.g., 3DES for ESP, HMAC-SHA-1-96 for AH) with its associated attributes (e.g., hash length).

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

to generate a session key and is dependent on the key exchange algorithm used.

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

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

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

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

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

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

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

ISAKMP Exchanges

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

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

(a) Base Exchange

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

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

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

(b) Identity Protection Exchange

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

The first two messages establish the SA. The next two messages perform key exchange, with nonces for replay protection. Once the session key has been computed, the two parties exchange encrypted messages that contain authentication information, such as digital signatures and optionally certificates validating the public keys.

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

(c) Authentication Only Exchange

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

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

4. Aggressive Exchange: minimizes the number of exchanges at the expense of not providing identity protection.

(d) Aggressive Exchange

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

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

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

(e) Informational Exchange

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

UNIT-V

Web Security: *Web Security Considerations, Secure Socket Layer (SSL) and Transport Layer Security (TLS), Secure Electronic Transaction (SET).* ***Intruders, Viruses and Firewalls:*** *Intruders, Intrusion Detection, Password Management, Virus and related threats, Countermeasures, Firewall Design Principles, Types of Firewalls.* ***Case Studies on Cryptography and Security****: Secure Inter Branch Transactions, Cross Site Vulnerability, Virtual Elections.*

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

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

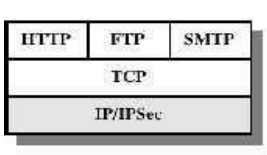
A summary of types of security threats faced in using the Web is given below:

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

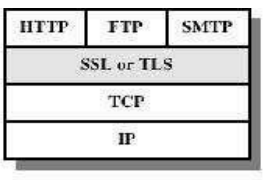
One way of grouping the security threats is in terms of passive and active attacks. *Passive attacks* include eavesdropping on network traffic between browser and server and gaining access to information on a website that is supposed to be restricted. *Active attacks* include impersonating another user,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

Web Traffic Security Approaches

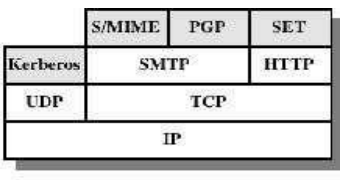
Various approaches for providing Web Security are available, where they are similar in the services they provide and also similar to some extent in the mechanisms they use. They differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack. The main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



(a) Network Level



(b) Transport Level



(c) Application Level

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

IPSec provides security at the network level and the main advantage is that it is transparent to end users and applications. In addition, IPSec includes a filtering capability so that only selected traffic can be processed.

Secure Socket Layer or Transport Layer Security (SSL/TLS) provides security just above the TCP at transport layer. Two implementation choices are present here.

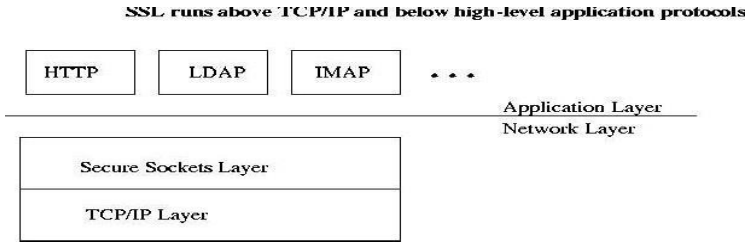
- Firstly, the SSL/TLS can be implemented as apart of TCP/IP protocol suite, thereby being transparent to applications.
- Alternatively, SSL can be embedded in specific packages like SSL being implemented by Netscape and Microsoft Explorer browsers.

Secure Electronic Transaction (SET) approach provides application-specific services i.e., according to the security requirements of a particular application. The main advantage of this approach is that service can be tailored to the specific needs of a given application.

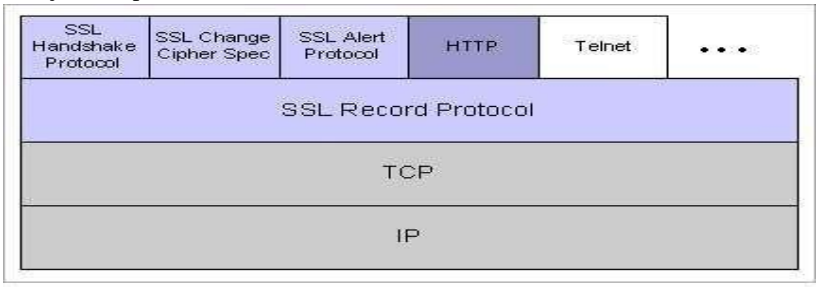
SECURE SOCKET LAYER/TRANSPORT LAYER SECURITY

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

SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity and encryption for privacy. SSL protocol has different versions such as SSLv2.0, SSLv3.0, where SSLv3.0 has an advantage with the addition of support for certificate chain loading. SSL 3.0 is the basis for the Transport Layer Security [TLS] protocol standard.



SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol, but rather two layers of protocols as shown below:



SSL Protocol Stack

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

The Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

- **Connection**: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
- **Session**: An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

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

- A session state is defined by the following parameters:
- ***Session identifier***: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- ***Peer certificate:***An X509.v3 certificate of the peer. This element of the state may be null.
- ***Compression method***: The algorithm used to compress data prior to encryption.
- ***Cipher spec:***Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.
- ***Master secret:***48-byte secret shared between the client and server.
- ***Is resumable:***A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters:

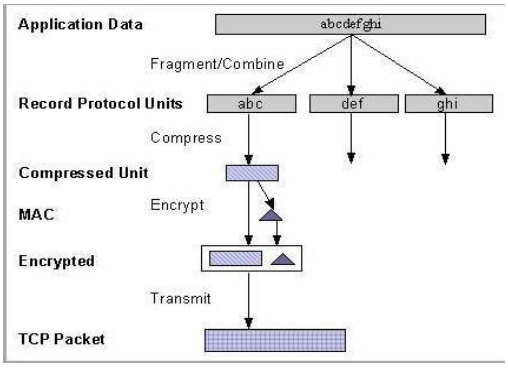
- ***Server and client random***: Byte sequences that are chosen by the server and client for each connection.
- ***Server write MAC secret***: The secret key used in MAC operations on data sent by the server.
- ***Client write MAC secret***: The secret key used in MAC operations on data sent by the client.
- ***Server write key:***The conventional encryption key for data encrypted by the server and decrypted by the client.
- ***Client write key***: The conventional encryption key for data encrypted by the client and decrypted by the server.
- ***Initialization vectors*:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV with the following record.
- ***Sequence numbers***: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264-1.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

- Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

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



The first step is fragmentation. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, compression is optionally applied.

Compression must be lossless and may not increase the content length by more than 1024 bytes. The next step in processing is to compute a message authentication code over the compressed data. For this purpose, a shared secret key is used.

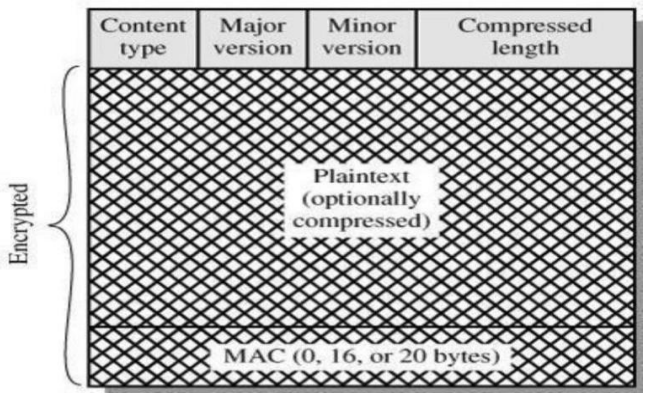
The calculation is defined as:

hash(MAC_write_secret || pad_2 || hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

Where, MAC_write_secret = Secret shared key pad_1 = the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times for pad_2 = the byte 0x5C (01011100) repeate d 48 times for MD5 and 40 times for SHA- 1

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

SSL Record Format



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

- Content Type (8 bits): The higher layer protocol used to process the enclosed fragment.
- Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.
- Minor Version (8 bits): Indicates minor version in use. For SSLv3, the value is 0.
- Compressed Length (16 bits): The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is 214 + 2048.

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

SSL Change Cipher Spec Protocol

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

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

SSL Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes. The first byte takes the value warning(1) or fatal(2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the

same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. The fatal alerts are listed below

- unexpected_message: An inappropriate message was received. bad_record_mac: An incorrect MAC was received.
- decompression_failure: The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowablelength).
- handshake_failure: Sender was unable to negotiate an acceptable set of security parameters given the options available.
- illegal_parameter: A field in a handshake message was out of range or inconsistent with other fields.

The remainder of the alerts are given below:

- close_notify: Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.
- no_certificate: May be sent in response to a certificate request if no appropriate certificate is available.
- bad_certificate: A received certificate was corrupt (e.g., contained a signature that did not verify).
- unsupported_certificate: The type of the received certificate is not supported. certificate_revoked: A certificate has been revoked by its signer. certificate_expired: A certificate has expired.
- certificate_unknown: Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

SSL Handshake Protocol

- SSL Handshake protocol ensures establishment of reliable and secure session between client and server and also allows server & client to:
- authenticate each other
- to negotiate encryption & MAC algorithms
- to negotiate cryptographic keys to b eused
- The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown below and each message has three fields:
- **Type (1 byte**): Indicates one of 10 messages.

(c) Handshake Protocol

- **Length (3 bytes):** The length of the message in bytes.
- **Content (>=0 bytes):** The parameters associated with this message

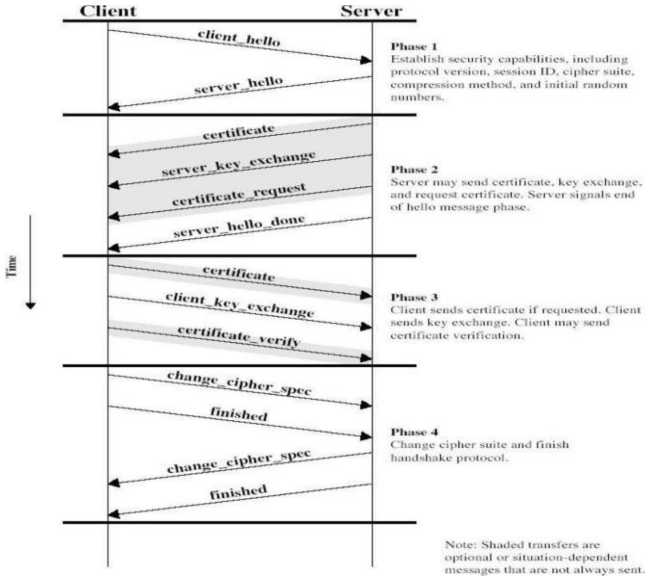
The following figure shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.in phases o Establish Security Capabilities

- o Server Authentication and Key Exchange
- o Client Authentication and Key Exchange
- o Finish

Phase 1. Establish Security Capabilities

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

- Version: The highest SSL version understood by the client.
- Random: A client-generated random structure, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as N onces and are used during key exchange to prevent replay attacks.



Phase 2. Server Authentication and Key Exchange

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

(1) The server has sent a certificate with fixed Diffie-Hellman parameters, or

(2) RSA key exchange is to be used.

Phase 3. Client Authentication and Key Exchange

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

Phase 4. Finish

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

Transport Layer Security

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

- Version Number: The TLS Record Format is the same as that of the SSL Record Format and the

fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 1.

Message Authentication Code:-

There are two differences between the SSLv3 and TLS MAC schemes: the actual algorithm and the scope of the MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104.HMAC is defined as follows:

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

where

H	=	embedded hash function (for TLS, either MD5 or SHA-1)
M	=	message input to HMAC
K^+	=	secret key padded with zeros on the left so that the result is equal to the block length of the hash code(for MD5 and SHA-1, block length = 512 bits)
ipad	=	00110110 (36 in hexadecimal) repeated 64 times (512 bits)
opad	=	01011100 (5C in hexadecimal) repeated 64 times (512 bits)

- SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. The level of security should be about the same in both cases.

For TLS, the MAC calculation encompasses the fields indicated in the following expression:

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

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

Pseudorandom Function:-TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs.

P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) || HMAC_hash(secret, A(2) || seed) || HMAC_hash(secret, A(3) || seed) || ...
Where A() is defined as
A(0) = seed
A(i) = HMAC_hash (secret, A(i - 1))

The data expansion function makes use of the HMAC algorithm, with either MD5 or SHA-1 as the underlying hash function. As can be seen, P_hash can be iterated as many times as necessary to produce the required quantity of data. For example, if P_SHA-1 was used to generate 64 bytes of data, it would have to be iterated four times, producing 80 bytes of data, of which the last 16 would be discarded. In this case, P_MD5 would also have to be iterated four times, producing exactly 64 bytes of data.

SET (SECURE ELECTRONIC TRANSACTION)

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

- Provides a secure communications channel among all parties involved in a transaction
- Provides trust by the use of X.509v3 digital certificates
- Ensures privacy because the information is only available to parties in a transaction when and where necessary

SET Requirements: lists the following business requirements for secure payment processing with credit cards over the Internet and other networks:

- **Provide confidentiality of payment and ordering information**: It is necessary to assure cardholders that this information is safe and accessible only to the intended recipient. Confidentiality also reduces the risk of fraud by either party to the transaction or by malicious third parties. SET uses encryption to provide confidentiality.
- **Ensure the integrity of all transmitted data**: That is, ensure that no changes in content occur during transmission of SET messages. Digital signatures are used to provide integrity.
- **Provide authentication that a cardholder is a legitimate user of a credit card account**: A mechanism that links a cardholder to a specific account number reduces the incidence of fraud and the overall cost of payment processing. Digital signatures and certificates are used to verify that a cardholder is a legitimate user of a valid account.
- **Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution:**This is the complement to the preceding requirement. Cardholders need to be able to identify merchants with whom they can conduct secure transactions. Again, digital signatures and certificates are used.
- **Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction:**SET is a well-tested specification based on highly secure cryptographic algorithms and protocols.
- **Create a protocol that neither depends on transport security mechanisms nor prevents their use**: SET can securely operate over a "raw" TCP/IP stack. However, SET does not interfere with the use of other security mechanisms, such as IPsec and SSL/TLS.
- **Facilitate and encourage interoperability among software and network providers**: The SET protocols and formats are independent of hardware platform, operating system, and Web software.

Key Features of SET

SET incorporates the following features:

- **Confidentiality of information**:Cardholder account and payment information is secured as it travels across the network. An interesting and important feature of SET is that it prevents the merchant from learning the cardholder's credit card number.
- **Integrity of data**:Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions. SET guarantees that these message contents are not altered in transit.
- **Cardholder account authentication**: SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.
- **Merchant authentication**: SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards. SET uses X.509v3 digital certificates with RSA signatures for this purpose

SET Participants

Cardholder: purchasers interact with merchants from personal computers over the Internet

Merchant: a person or organization that has goods or services to sell to the cardholder

Issuer: a financial institution, such as a bank, that provides the cardholder with the payment card.

Acquirer: a financial institution that establishes an account with a merchant and processes payment card Authorizations and payments

Payment gateway: a function operated by the acquirer or a designated third party that processes merchant payment messages

Certification authority (CA): an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways

Events in a transaction

1. The customer obtains a credit card account with a bank that supports electronic payment and SET
2. The customer receives a X.509v3 digital certificate signed by the bank.
3. Merchants have their own certificates
4. The customer places an order.

5. The merchant sends a copy of its certificate so that the customer can verify that it's a valid store
6. The order and payment are sent
7. The merchant requests payment authorization
8. The merchant confirms the order
9. The merchant ships the goods or provides the service to the customer
10. The merchant requests payment

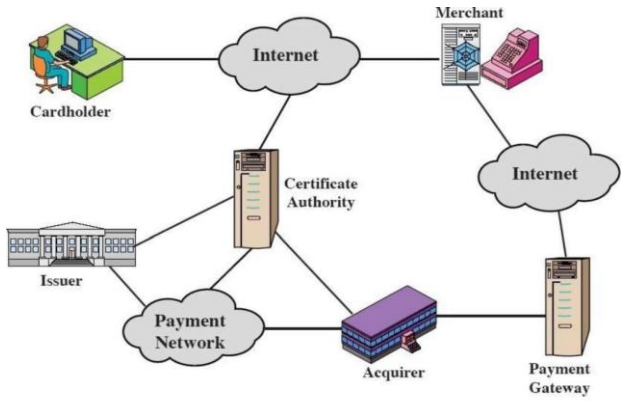
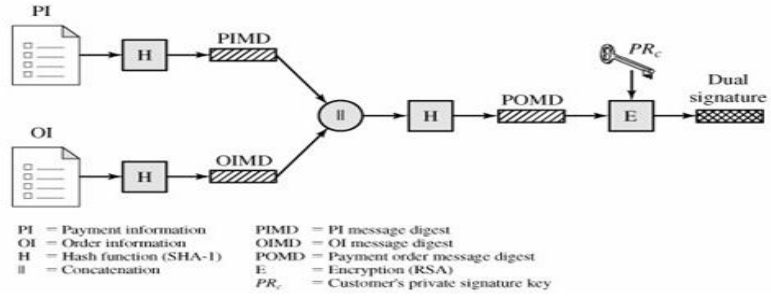


Figure 17.8 Secure Electronic Commerce Components

Dual Signature

Introduction:-The purpose of the dual signature is to link two messages that are intended for two different recipients. If the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate

Suppose that the customers send the merchant two messages: a signed OI and a signed PI, and the merchant passes the PI on to the bank. If the merchant can capture another OI from this customer, the merchant could claim that this OI goes with the PI rather than the original OI.



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

$$DS = E (PR_c, [H (H (PI) || H (OI)])$$

Where PR_c is the customer's private signature key. Now suppose that the merchant is in possession of the dual signature (DS), the OI, and the message digest for the PI (PIMD). The merchant also has the public key of the customer, taken from the customer's certificate. Then the merchant can compute the quantities

$$H (PIMS||H [OI]); D (PU_c, DS)$$

Where PU_c is the customer's public signature key. If these two quantities are equal, then the merchant has verified the signature. Similarly, if the bank is in possession of DS, PI, the message digest for OI (OIMD), and the customer's public key, then the bank can compute

$$H (H [OI] ||OIMD); D (PU_c, DS)$$

If these two quantities are equal, then the bank has verified the signature. In summary,

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

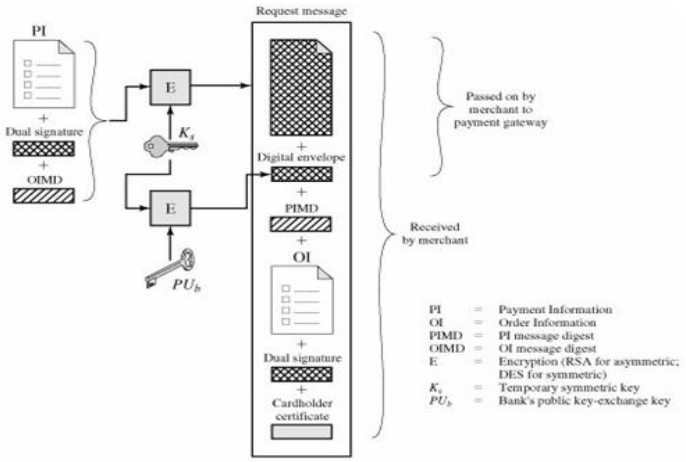
Payment Processing

Description:-Payment Processing includes the following transactions:-

- Purchase request
- Payment authorization
- Payment capture

Purchase Request:-The purchase request exchange consists of four messages: Initiate Request, Initiate Response, Purchase Request, and Purchase Response.

1. The customer requests the certificates in the **Initiate Request** message, sent to the merchant. This message includes the brand of the credit card that the customer is using. The message also includes an ID assigned to this request/response pair by the customer and a nonce used to ensure timeliness.
2. The merchant generates a response and signs it with its private signature key. The response includes the nonce from the customer, another nonce for the customer to return in the next message, and a transaction ID for **this purchase transaction**. In addition to the signed response, the Initiate Response message includes the merchant's signature certificate and the payment gateway's key exchange certificate.



The cardholder verifies the merchant and gateway certificates by means of their respective CA signatures and then creates the OI and PI. The transaction ID assigned by the merchant is placed in both the OI and PI. The OI does not contain explicit order data such as the number and price of items. Rather, it contains an order reference generated in the exchange between merchant and customer during the shopping phase before the first SET message.

When the merchant receives the Purchase Request message, it performs the following actions:

- Verifies the cardholder certificates by means of its CA signatures.
- Verifies the dual signature using the customer's public signature key. Processes the order and forwards the payment information to the payment gateway for authorization.
- Sends a purchase response to the cardholder

Payment Authorization:-During the processing of an order from a cardholder, the merchant authorizes the transaction with the payment gateway. The payment authorization ensures that the transaction was approved by the issuer. This authorization guarantees that the merchant will receive payment; the merchant can therefore provide the services or goods to the customer.

Payment Capture:-For the Capture Request message, the merchant generates, signs, and encrypts a capture request block, which includes the payment amount and the transaction ID**.**The gateway then notifies the merchant of payment in a Capture Response message. The message includes a capture response block that the gateway signs and encrypts.

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 misuses 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.

Intrusion Techniques:-The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system.Generally this information is in the form of a user password. With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user.The password file can be protected in one of two ways:

One-way function: The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value.

Access control: Access to the password file is limited to one or a very few accounts.

The following techniques are for learning passwords:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords (those of one to three characters).
3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.

5. Try users' phone numbers, Social Security numbers, and room numbers.

6. Try all legitimate license plate numbers for this state.

7. Use a Trojan horse to bypass restrictions on access.

8. Tap the line between a remote user and the host system.

COUNTERMEASURES:

- Detection concerned with learning of an attack, either before or after its success.
- Prevention challenging security goal and an uphill bottle at all times

The first six methods are various ways of guessing a password. If an intruder has to verify the guess by attempting to log in, it is a tedious and easily countered means of attack.The seventh method of attack, the Trojan horse, can be particularly difficult to counter.The eighth attack, line tapping, is a matter of physical security. It can be countered with link encryption techniques.

Other intrusion techniques do not require learning a password. Intruders can get access to a system by exploiting attacks such as buffer overflows on a program that runs with certain privileges.

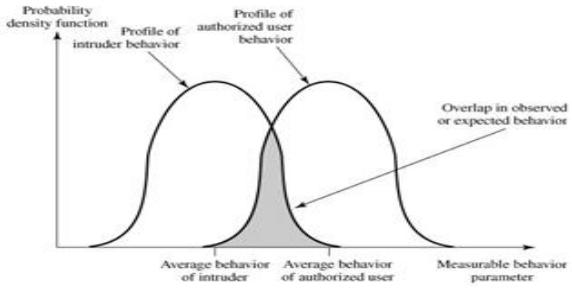
Intrusion Detection:

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:

1. 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 arecompromised.
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

It is assumed that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified. 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.

Behavior of Intruders and Authorized



Approaches to intrusion detection:

1. **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.

2. **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.

Statistical anomaly detection is effective against masqueraders, who are unlikely to mimic the behavior patterns of the accounts they appropriate. 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.

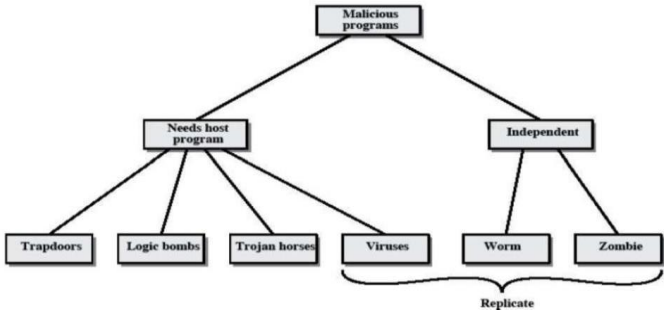
Audit Records:-A fundamental tool for intrusion detection is the audit record. Basically, two plans are used:

- **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.

VIRUSES AND RELATED THREATS

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

Malicious Programs



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
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

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.

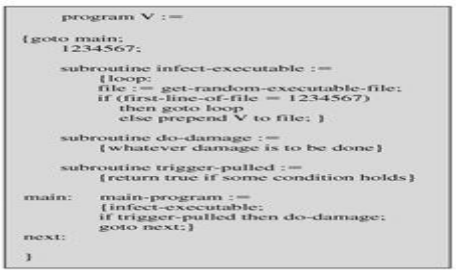
The Nature of Viruses

Introduction:-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.

The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network.

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.

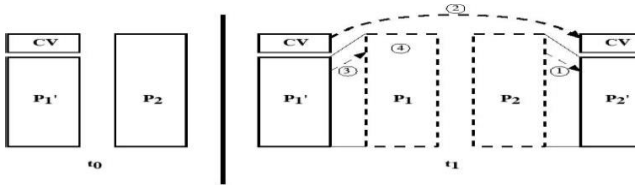


```
program V :=
{goto main;
      1234567;
      subroutine infect-executable :=
            {loop:
            file := get-random-executable-file;
            if (first-line-of-file = 1234567)
                  then goto loop
                  else prepend V to file; }
      subroutine do-damage :=
            {whatever damage is to be done}
      subroutine trigger-pulled :=
            {return true if some condition holds}
main:     main-program :=
            {infect-executable;
            if trigger-pulled then do-damage;
            goto next;}
next:
}
```

Virus Structure:-A virus can be prepended or postponed 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.The virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line 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.
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, and Figure 19.3 [COHE94] illustrates the operation. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

- For each uninfected file P2 that is found, the virus first compresses that file to produce P'2, which is shorter than the original program by the size of thevirus.
- A copy of the virus is prepended to the compressed program.
- The compressed version of the original infected program, P'1, is uncompressed The
- uncompressed original program is executed.



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.

Types of Viruses

Introduction:-The following categories as being among the most significant types of viruses:

- **Parasitic virus**: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.
- **Memory-resident virus**: Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.
- **Boot sector virus**: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
- **Stealth virus**: A form of virus explicitly designed to hide itself from detection by antivirus software.

- **Polymorphic virus**: A virus that mutates with every infection, making detection by the "signature" of the virus impossible.
- **Metamorphic virus**: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.
- **Sealth virus** is a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program. Thus, stealth is not a term that applies to a virus as such but, rather, is a technique used by a virus to evade detection.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns. As with a stealth virus, the purpose is to defeat programs that scan for viruses. In this case, the "signature" of the virus will vary with each copy. To achieve this variation, the virus may randomly insert superfluous instructions or interchange the order of independent instructions. A more effective approach is to use encryption. A portion of the virus, generally called a mutation engine, creates a random encryption key to encrypt

the remainder of the virus. The key is stored with the virus, and the mutation engine itself is altered. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected.

Macro Viruses:-Macro viruses are particularly threatening for a number of reasons

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.
2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro.

E-mail Viruses:-A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
2. The virus does local damage.

The virus propagates itself as soon as activated to all of the e-mail addresses known to the infected host. As a result, whereas viruses used to take months or years to propagate, they now do so in hours. This makes it very difficult for antivirus software to respond before much damage is done

Worms

Introduction:-A worm is a program that can replicate itself and send copies from computer to computer across network connections. The worm may be activated to replicate and propagate again. In addition to propagation, the worm usually performs some unwanted function. An e-mail virus has some of the characteristics of a worm, because it propagates itself from system to system.

Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

Examples include the following:

- **Electronic mail facility**: A worm mails a copy of itself to other systems.
- **Remote execution capability**: A worm executes a copy of itself on another system.
- **Remote login capability**: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally performs the following functions:

1. Search for other systems to infect by examining host tables or similar repositories of remote system addresses.
2. Establish a connection with a remote system.
3. Copy itself to the remote system and cause the copy to be run.

The network worm may also attempt to determine whether a system has previously been infected before copying itself to the system. In a multiprogramming system, it may also disguise its presence by naming itself as a system process or using some other name that may not be noticed by a system operator.As with viruses, network worms are difficult to counter.

State of Worm Technology

The state of the art in worm technology includes the following:

- **Multiplatform**: Newer worms are not limited to Windows machines but can attack a variety of platforms, especially the popular varieties of UNIX.
- **Multiexploit**: New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, andother network-based applications.

- **Ultrafast spreading**: One technique to accelerate the spread of a worm is to conduct a prior Internet scan to accumulate Internet addresses of vulnerable machines.
- **Polymorphic**: To evade detection, skip past filters, and foil real-time analysis, worms adopt the virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.
- **Metamorphic**: In addition to changing their appearance, metamorphic worms have a repertoire of behavior patterns that are unleashed at different stages of propagation.
- **Transport vehicles**: Because worms can rapidly compromise a large number of systems, they are ideal for spreading other distributed attack tools, such as distributed denial of service zombies.
- **Zero-day exploit**: To achieve maximum surprise and distribution, a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched.

The Morris Worm

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password- cracking program that tried

    a. Each user's account name and simple permutations of it

    b. A list of 432 built-in passwords that Morris thought to be likely candidates

    c. All the words in the local system directory

- It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.
- It exploited a trapdoor in the debug option of the remote process that receives and sends mail.
- If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter.

Recent Worm Attacks In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms:

- from client to client via e-mail
- from client to client via open network shares
- from Web server to client via browsing of compromised Web sites
- from client to Web server via active scanning for and exploitation of various Microsoft.

Firewall design principles:

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

1. Defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
2. provides a location for monitoring security-related events
3. is a convenient platform for several Internet functions that are not security related, such as NAT and Internet usage audits or logs
4. A firewall can serve as the platform for IPSec to implement virtual private networks.

Following are the design goals for a firewall:-

1. 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.
2. 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, as explained later in this section.

3. The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system.

Originally, firewalls focused primarily on service control, but they have since evolved to provide all four:

- **Service control**: Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the 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 a Web or mail service.
- **Direction control**:Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.
- **User control**:Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter. It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology.
- **Behavior control**:Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

The following capabilities are within the scope of a firewall:

1. 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. The use of a single choke point simplifies security management because security capabilities are consolidated on a single system or set of systems.

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

3. A firewall is a convenient platform for several Internet functions that are not security related. These include a network address translator, which maps local addresses to Internet addresses, and a network management function that audits or logs Internet usage.

4. A firewall can serve as the platform for IPSec.

Firewalls have their limitations, including the following:

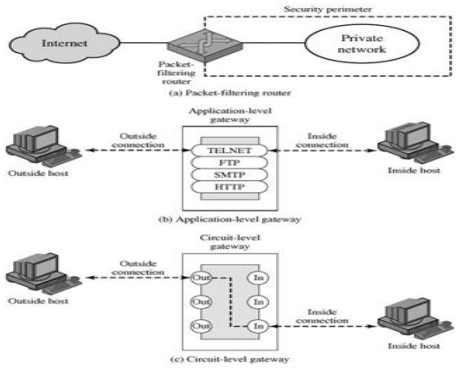
1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.

2. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.

3. 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

Introduction: -The three common types of firewalls: packet filters, application-level gateways, and circuit-level gateways.



Packet-Filtering Router:-A packet-filtering router applies a set of rules to each incoming and outgoing 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 information contained in a network packet:-

- **Source IP address**: The IP address of the system that originated the IP packet.
- **Destination IP address**: The IP address of the system the IP packet is trying to reach.
- **Source and destination transport-level address**: The transport level.
- **IP protocol field**: Defines the transport protocol.
- **Interface**: For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. Two default policies are possible:

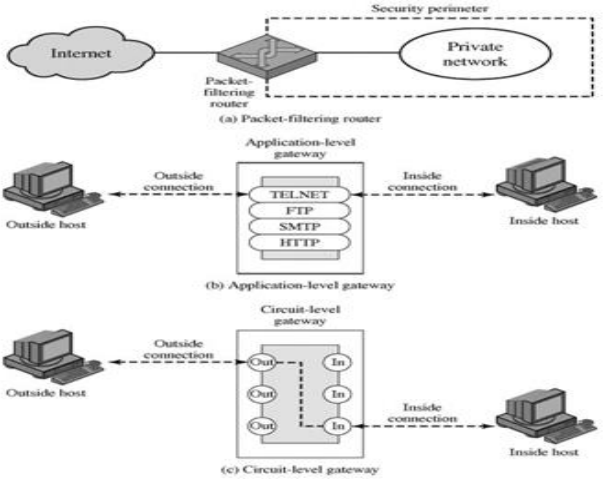
- Default = discard: That which is not expressly permitted is prohibited.
- Default = forward: That which is not expressly prohibited is permitted.

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. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.

Circuit-Level Gateway:-A third type of firewall is the circuit-level gateway. This can be a stand-alone system or it can be a specialized 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 outside 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. In this configuration, the gateway can incur the processing overhead of examining incoming application data for forbidden functions but does not incur that overhead on outgoing data.

Firewall Configurations:

Description:-The three common firewall configurations:-



The firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.
2. For traffic from the internal network, only IP packets from the bastion host are allowed out.

The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application-level gateway alone, for two reasons. First, this configuration implements both packet-level and application-level filtering, allowing for considerable flexibility in defining security policy. Second, an intruder must generally penetrate two separate systems before the security of the internal network is compromised.

This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet.

The screened subnet firewall configuration is the most secure of those we have considered. In this configuration, two packet-filtering routers are used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages:

1. There are now three levels of defense to thwart intruders.
2. The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet.
3. Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet.

Case Studies on Cryptography and security: Secure Inter-Branch Payment Transactions

Securing inter-branch payment transactions is crucial for maintaining the integrity, confidentiality, and authenticity of financial transactions within an organization. Here are some best practices and measures to enhance the security of inter-branch payment transactions:

Encryption:

- Use strong encryption protocols, such as TLS (Transport Layer Security), to encrypt data during transmission between branches.
- Ensure that all communication channels, including online platforms and internal networks, use encryption to protect sensitive information from interception.

Multi-Factor Authentication (MFA):

- Implement multi-factor authentication for accessing financial systems and conducting transactions.

- This adds an extra layer of security by requiring users to verify their identity through multiple means, such as passwords, biometrics, or smart cards.

Access Control:

- Enforce strict access controls to limit the access of employees to only the resources and systems necessary for their roles.
- Regularly review and update access permissions based on job roles, and promptly revoke access for employees who no longer require it.

Secure Networks:

- Ensure that internal networks connecting different branches are secure and segmented to prevent unauthorized access. Utilize firewalls and intrusion detection/prevention systems to monitor and protect the network.

Regular Audits and Monitoring:

- Conduct regular audits of financial transactions to identify any unusual or suspicious activities.
- Implement real-time monitoring systems to detect and respond to potential security incidents promptly.

Secure Endpoints:

- Ensure that all endpoints (computers, tablets, etc.) used for financial transactions are secure and up-to-date with the latest security patches.
- Use endpoint protection software and conduct regular security scans.

Employee Training:

- Train employees on security best practices, including how to recognize phishing attempts, the importance of strong passwords, and the proper handling of sensitive information.
- Foster a culture of security awareness within the organization.

Secure File Transfer:

- Use secure methods for transferring financial data between branches. Implement secure file transfer protocols, and consider using secure file transfer applications.

Vendor Security:

- If third-party vendors are involved in payment processing or financial systems, ensure they adhere to strict security standards. Perform due diligence on vendors and regularly assess their security practices.

Incident Response Plan:

- Develop and regularly update an incident response plan to guide the organization's response in the event of a security incident.
- This should include procedures for identifying, containing, eradicating, recovering, and learning from security breaches.

Regular Software Updates:

- Keep all software, including financial applications and security tools, up-to-date with the latest patches and updates to address known vulnerabilities.

Data Backups:

- Regularly back up critical financial data and ensure that the backup process is secure. This helps in recovering data in case of any security incidents or system failures.

Implementing a comprehensive security strategy that encompasses these measures will significantly enhance the security of inter-branch payment transactions and protect the financial integrity of the organization. Regularly reassess and update security measures to adapt to evolving threats and technologies.

Cross-Site Scripting (XSS):

Cross-Site Scripting (XSS) is a web security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can be executed in the context of a user's browser, leading to potential security risks. There are several types of XSS vulnerabilities, but the common goal is to execute unauthorized scripts in a user's browser.

Here are some general guidelines to prevent and mitigate XSS vulnerabilities:

Input Validation:

- Validate and sanitize all user inputs on the server side. Ensure that user inputs conform to expected formats and reject any input that contains malicious scripts.

Output Encoding:

- Encode user-generated content before displaying it in the browser. This helps to ensure that any potentially dangerous characters are displayed as literals rather than being executed as scripts.

Content Security Policy (CSP):

- Implement a Content Security Policy to restrict the types of content that can be executed on a webpage. A properly configured CSP can mitigate the impact of XSS attacks by controlling which scripts can be executed.

HTTP-only Cookies:

- Set the "HttpOnly" flag on cookies to prevent them from being accessed through JavaScript. This helps protect sensitive session information from being stolen through XSS attacks.

Secure Flag for Cookies:

- Set the "Secure" flag on cookies to ensure they are only transmitted over secure, encrypted connections (HTTPS). This helps prevent attackers from intercepting sensitive information during transit.

Update and Patch Libraries:

- Keep all web application frameworks, libraries, and third-party components up-to-date. Developers should regularly check for security updates and apply patches to address known vulnerabilities.

Avoid Inline JavaScript:

- Minimize the use of inline JavaScript within HTML pages. Instead, use external JavaScript files and link to them securely. This reduces the attack surface for XSS vulnerabilities.

Use Frameworks and Libraries Securely:

- If using JavaScript frameworks or libraries, follow best practices for secure coding. Many frameworks offer features to help prevent XSS, so be sure to leverage those features correctly.

Educate Developers:

- Train developers on secure coding practices and make them aware of the risks associated with XSS vulnerabilities. Code reviews and static analysis tools can also help identify potential security issues.

Web Application Firewall (WAF):

- Deploy a Web Application Firewall to help detect and block malicious traffic, including attempts to exploit XSS vulnerabilities. A WAF can provide an additional layer of defense against various web-based attacks.

Regular Security Audits:

- Conduct regular security audits and penetration testing to identify and address potential vulnerabilities in the application. This can help discover and fix issues before they can be exploited by attackers.

By implementing these measures, web developers and administrators can significantly reduce the risk of XSS vulnerabilities and enhance the overall security of their web applications. Regularly reviewing and updating security practices is crucial, as new threats and techniques can emerge over time.

Virtual Elections System Architecture:

User Interface:

- **Voter Interface:** A secure and user-friendly interface for voter registration, authentication, and casting votes. This could be a web application or a mobile app.

Frontend:

- **Voter Registration Module:** Allows users to register as voters by providing necessary information.
- **Candidate Registration Module:** Enables individuals to register as candidates for different positions.
- **Authentication and Authorization:**
- **User Authentication:** Secure authentication mechanisms to ensure the identity of voters and candidates.
- **Authorization:** Role-based access control to manage permissions for different user roles.

Voting System:

- **Secure Voting Module:** Provides a platform for voters to cast their votes securely. This should include cryptographic methods to ensure the integrity and confidentiality of votes.
- **Voting Period Management:** Controls the timing of the election phases, such as the campaign period and the actual voting period.

Candidate Management:

- **Candidate Dashboard:** Allows candidates to manage their campaign information, monitor progress, and view results.
- **Campaign Period Control:** Manages the timeline for candidate campaigns.

Back-end Server:

- **User Management:** Manages voter and candidate data, including registration details and authentication.
- **Vote Management:** Collects, stores, and processes votes securely.
- **Results Calculation:** Calculates and verifies election results.
- **Security Measures:** Implements security protocols to protect against attacks, including encryption, firewalls, and intrusion detection systems.

Database:

- **Voter Database:** Stores information about registered voters.
- **Candidate Database:** Stores information about candidates.
- **Vote Database:** Securely stores cast votes with proper encryption.

Blockchain Integration (Optional):

- **Distributed Ledger**: Provides an immutable record of votes to enhance transparency and integrity.
- **Smart Contracts:** Automates certain aspects of the election process, such as vote counting.

External Systems Integration:

- **Identity Verification Services:** Integrates with identity verification services for additional authentication.
- **Notification Services:** Sends notifications to users about important events in the election process.

Security Measures:

- **SSL/TLS Encryption:** Ensures secure communication between clients and servers.
- **Multi-Factor Authentication (MFA):** Adds an extra layer of security for user authentication.

Regular Security Audits: Conducts regular security audits and vulnerability assessments.

Results Presentation:

- **Results Dashboard:** Displays election results to the public in real-time (if permissible) or after the voting period ends.

Audit Trail:

- **Audit Logging:** Maintains detailed logs of all activities for auditing purposes.

Compliance and Regulations:

- **Compliance Module:** Ensures that the virtual election system complies with relevant electoral laws and regulations.

Disaster Recovery and Redundancy:

- **Data Backup:** Regularly backs up critical data to prevent data loss.
- **Redundancy:** Implements redundant systems to ensure continuous operation in case of system failures.