PRATHYUSHA

ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

REGULATION 2017

IV YEAR - VII SEMESTER

CS8792 CRYPTOGRAPHY AND NETWORK SECURITY

CS8792 CRYPTOGRAPHY AND NETWORK SECURITY L T P C 3 0 0 3

OBJECTIVES: • To understand Cryptography Theories, Algorithms and Systems. • To understand necessary Approaches and Techniques to build protection mechanisms in order to secure computer networks

UNIT I INTRODUCTION 9

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

UNIT II SYMMETRIC KEY CRYPTOGRAPHY 9

MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY: Algebraic structures - Modular arithmetic- Euclid‟s algorithm- Congruence and matrices - Groups, Rings, Fields- Finite fields- SYMMETRIC KEY CIPHERS: SDES – Block cipher Principles of DES – Strength of DES – Differential and linear cryptanalysis - Block cipher design principles – Block cipher mode of operation – Evaluation criteria for AES – Advanced Encryption Standard - RC4 – Key distribution.

UNIT III PUBLIC KEY CRYPTOGRAPHY 9

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY: Primes – Primality Testing – Factorization – Euler‘s totient function, Fermat‘s and Euler‘s Theorem - Chinese Remainder Theorem – Exponentiation and logarithm - ASYMMETRIC KEY CIPHERS: RSA cryptosystem – Key distribution – Key management – Diffie Hellman key exchange - ElGamal cryptosystem – Elliptic curve arithmetic- Elliptic curve cryptography.

UNIT IV MESSAGE AUTHENTICATION AND INTEGRITY 9

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC – SHA –Digital signature and authentication protocols – DSS- Entity Authentication: Biometrics, Passwords, Challenge Response protocols- Authentication applications - Kerberos, X.509

UNIT V SECURITY PRACTICE AND SYSTEM SECURITY 9

Electronic Mail security – PGP, S/MIME – IP security – Web Security - SYSTEM SECURITY: Intruders – Malicious software – viruses – Firewalls.

TOTAL 45 PERIODS

OUTCOMES:

At the end of the course, the student should be able to:

• Understand the fundamentals of networks security, security architecture, threats and vulnerabilities

- Apply the different cryptographic operations of symmetric cryptographic algorithms
- Apply the different cryptographic operations of public key cryptography
- Apply the various Authentication schemes to simulate different applications.
- Understand various Security practices and System security standards

TEXT BOOK:

1. William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006.
2. REFERENCES:
   1. C K Shyamala, N Harini and Dr. T R Padmanabhan: Cryptography and Network Security, Wiley India Pvt.Ltd
   2. BehrouzA.Foruzan, Cryptography and Network Security, Tata McGraw Hill 2007.

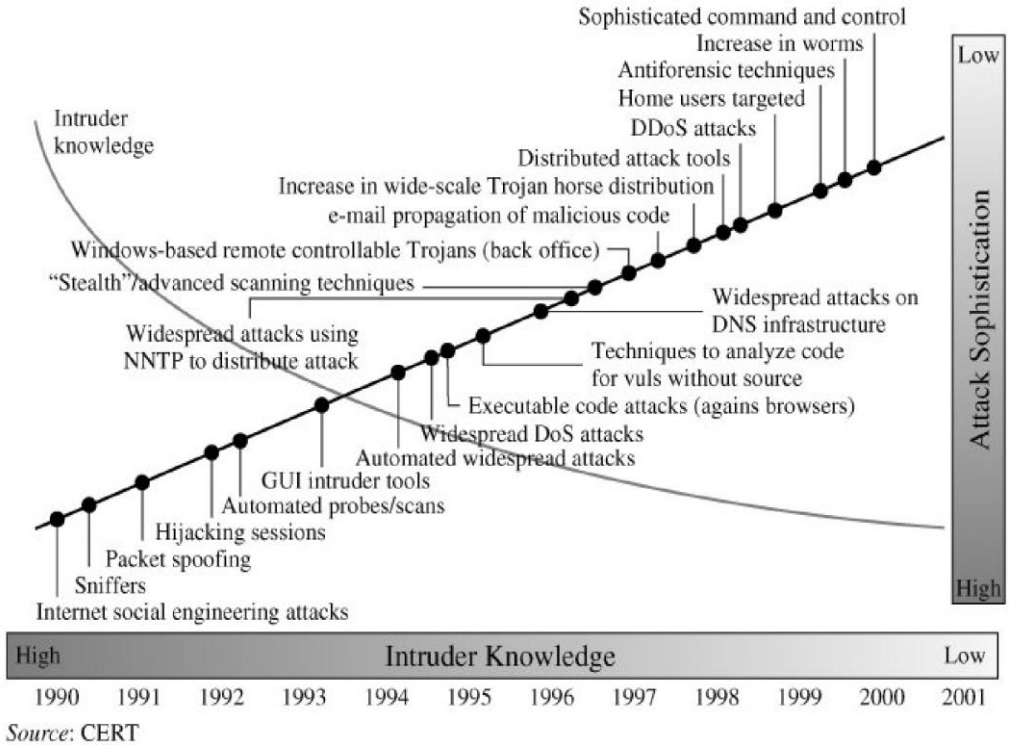
UNIT I

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

SECURITY TRENDS:

In 1994, the Internet Architecture Board (IAB) issued a report entitled "Security in the Internet Architecture" (RFC 1636). The report stated the general consensus that the Internet needs more and better security, and it identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms.

Figure 1.1. Trends in Attack Sophistication and Intruder Knowledge



Source: CERT

Important Terminologies

Plain text: An original message is known as the **plaintext**.

Cipher text: The coded message is called the **cipher text**.

Encryption: The process of converting from plaintext to cipher text is known as enciphering or encryption.

Decryption: The process of converting from cipher text in to plain text is known as deciphering or decryption.

Cryptography The many schemes used for encryption constitute the area of study known as cryptography. Such a scheme is known as a cryptographic system or a cipher.

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

Cryptology: The areas of cryptography and cryptanalysis together are called **cryptology**.

LEGAL, ETHICAL AND PROFESSIONAL ASPECTS OF SECURITY:

Laws:

Rules that mandate or prohibit certain behavior

Drawn from ethics

Ethics:

Define socially acceptable behaviors

Key difference:

Laws carry the authority of a governing body

Ethics do not carry the authority of a governing body

Cybercrime And Computer Crime:

- *Computer crime or cybercrime, is a term used broadly to* ***describe criminal activity*** **in which computers or computer networks are a tool, a target, or a place of criminal activity**.
- The term *cybercrime has a association of* the use of networks specifically, whereas *computer crime may or may not involve* networks.

Types of Computer Crime:

1)Computers as targets:

- **Acquire information** stored on that computer system to control the target system
- This form of crime involves an attack on data integrity, system integrity, data confidentiality, privacy, or availability

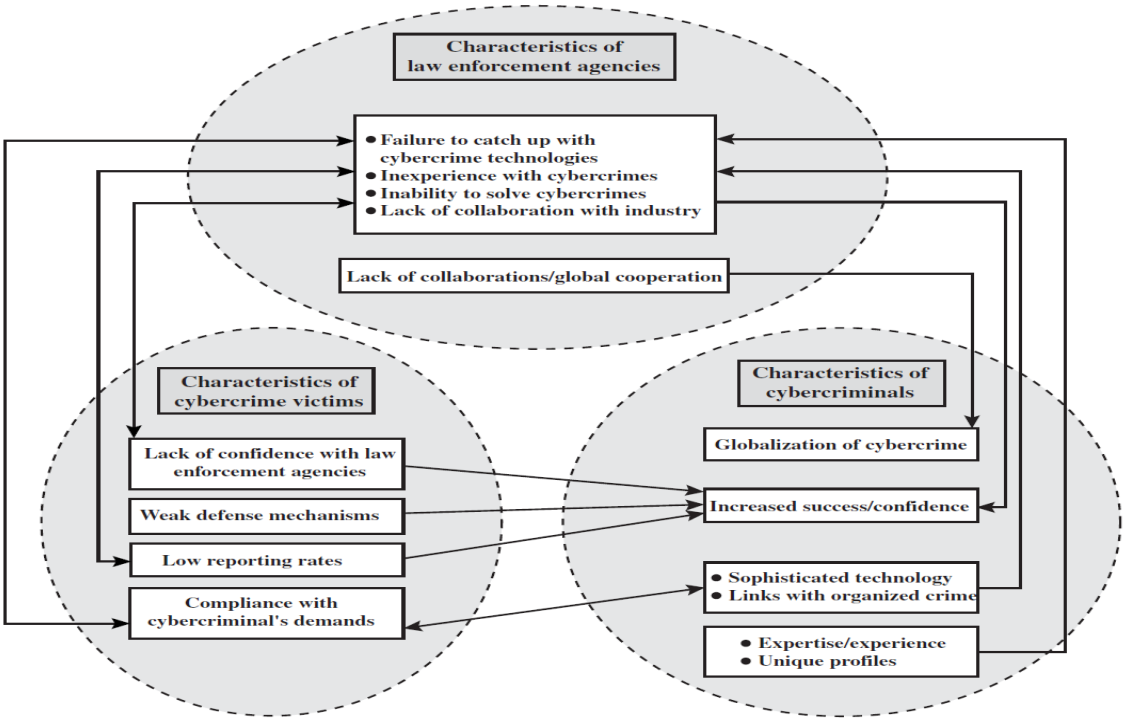
2)Computers as storage devices:

- Computers can be used to further unlawful activity by using a computer or a computer device as a passive storage medium.
- **Example**, the computer can be used to store stolen password lists, credit card or calling card numbers, proprietary corporate information

3)Computers as communications tools:

- Many of the crimes falling within this category are simply traditional crimes that are committed online.
- **Examples** include the illegal sale of prescription drugs, controlled substances and guns, fraud, gambling.

Law Enforcement Challenges:



Intellectual Property:

The U.S. legal system, and legal systems generally, distinguish three primary types of property:

Real property: Land and things permanently attached to the land, such as trees, buildings, and stationary mobile homes.

Personal property: Personal effects, moveable property and goods, such as cars, bank accounts, wages, securities, a small business, furniture, insurance policies, jewelry, patents, pets.

Intellectual property: Any intangible asset that consists of human knowledge and ideas. Examples include software, data, novels, sound recordings, the design of a new type of mousetrap, or a cure for a disease.

Types of Intellectual Property:

1. Copyrights
2. Trademarks
3. Patents

Copyrights:

- *Copyright law protects the tangible or fixed expression of an idea*
- A creator can claim copyright, and file for the copyright at a national government copyright office, if the following conditions are fulfilled:

- The proposed work is original.
- The creator has put this original idea into a concrete form, such as hard copy software, or multimedia form.
- Examples: Literary works, Musical works, Dramatic works, choreographic works, Architectural works, Software-related works.

The **copyright owner has the following exclusive rights**, protected against infringement:

- **Reproduction right:** Lets the owner make copies of a work
- **Modification right:** Also known as the derivative-works right, concerns modifying a work to create a new or derivative work
- **Distribution right:** Lets the owner publicly sell, rent, lease, or lend copies of the work.
- **Public-performance right:** Applies mainly to live performances
- **Public-display right:** Lets the owner publicly show a copy of the work directly or by means of a film, slide, or television image

PATENTS

A patent for an invention is the grant of a property right to the inventor.

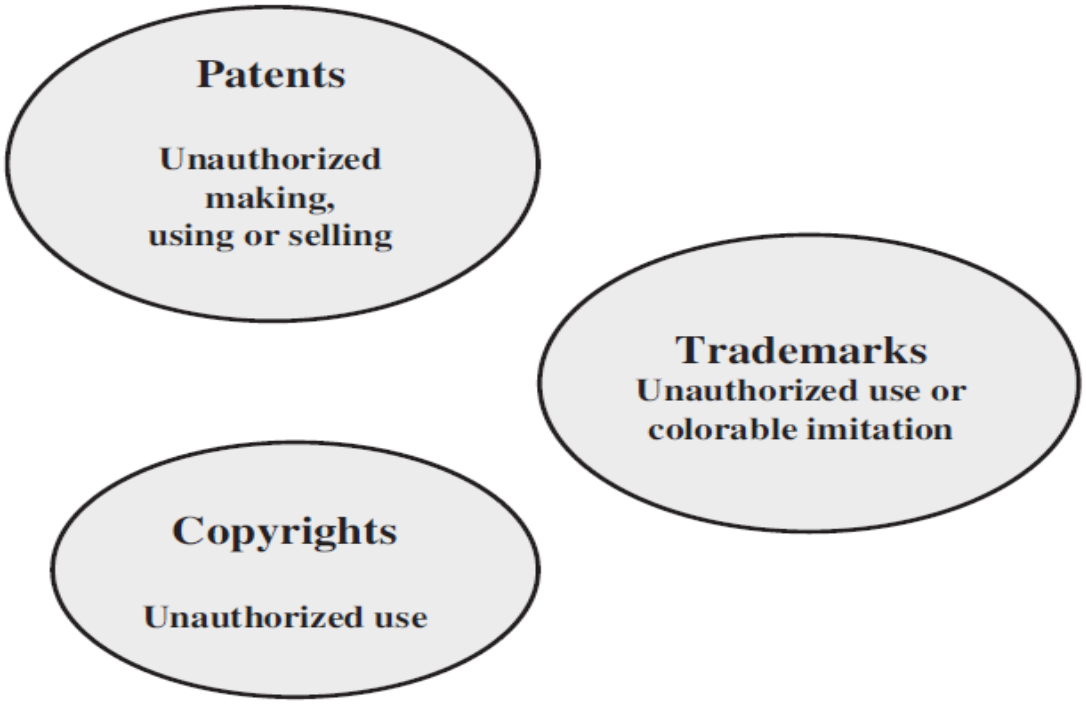
There are three types of patents:

1. Utility patents: May be granted to anyone who invents or discovers any new and useful process, machine, article of manufacture, or composition of matter, or any new and useful improvement thereof;
2. Design patents: May be granted to anyone who invents a new, original, and ornamental design for an article of manufacture;
3. Plant patents: May be granted to anyone who invents or discovers and reproduces any distinct and new variety of plant.

Trademarks:

A trademark is a word, name, symbol, or device that is used in trade with goods to indicate the source of the goods and to distinguish them from the goods of others.

Intellectual Property Infringement:



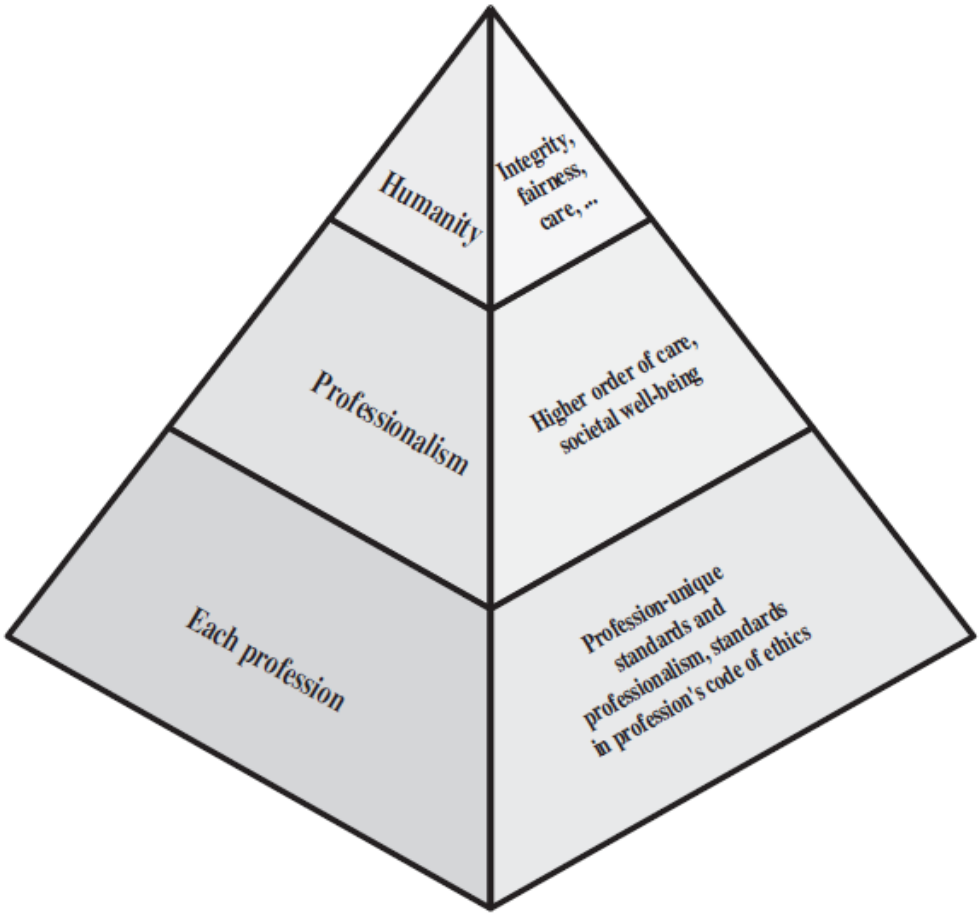
Intellectual Property Relevant to Network and Computer Security:

- Software
- Databases
- Digital content
- Algorithms

ETHICAL ISSUES:

Ethics refers to a system of moral principles that relates to the benefits and harms of particular actions, and to the rightness and wrongness of motives and ends of those actions.

The Ethical Hierarchy:



Ethical Issues related to Computers and Information security:

Ethical issues arise as the result of the roles of computers, such as the following:

- Repositories and processors of information
- Producers of new forms and types of assets
- Instruments of acts
- Symbols of intimidation and deception

Potential Ethical Dilemmas for Information Systems:

Technology Intrusion	**Privacy internal to the firm** **Privacy external to the firm** **Computer surveillance** **Employee monitoring** **Hacking**
Ownership Issues	**Proprietary rights** **Conflicts of interest** **Software copyrights** **Use of company assets for personal benefit** **Theft of data, software, or hardware**
Legal Issues and Social Responsibilities	**Embezzlement, fraud** **Accuracy and timeliness of data** **Over-rated system capabilities and "smart" computers** **Monopoly of data**
Personnel Issues	Employee sabotage Ergonomics and human factors Training to avoid job obsolescence

Digital Rights Management (DRM):

- Digital Rights Management (DRM) refers to systems and procedures that ensure that holders of digital rights are clearly identified and receive the stipulated payment for their works.
- The systems and procedures may also impose further restrictions on the use of digital objects, such as inhibiting printing or prohibiting further distribution.

DRM Objectives:

1.Provide persistent content protection against unauthorized access to the digital content, limiting access to only those with the proper authorization.

2. Support a variety of digital content types (e.g., music files, video streams, digital books, images).

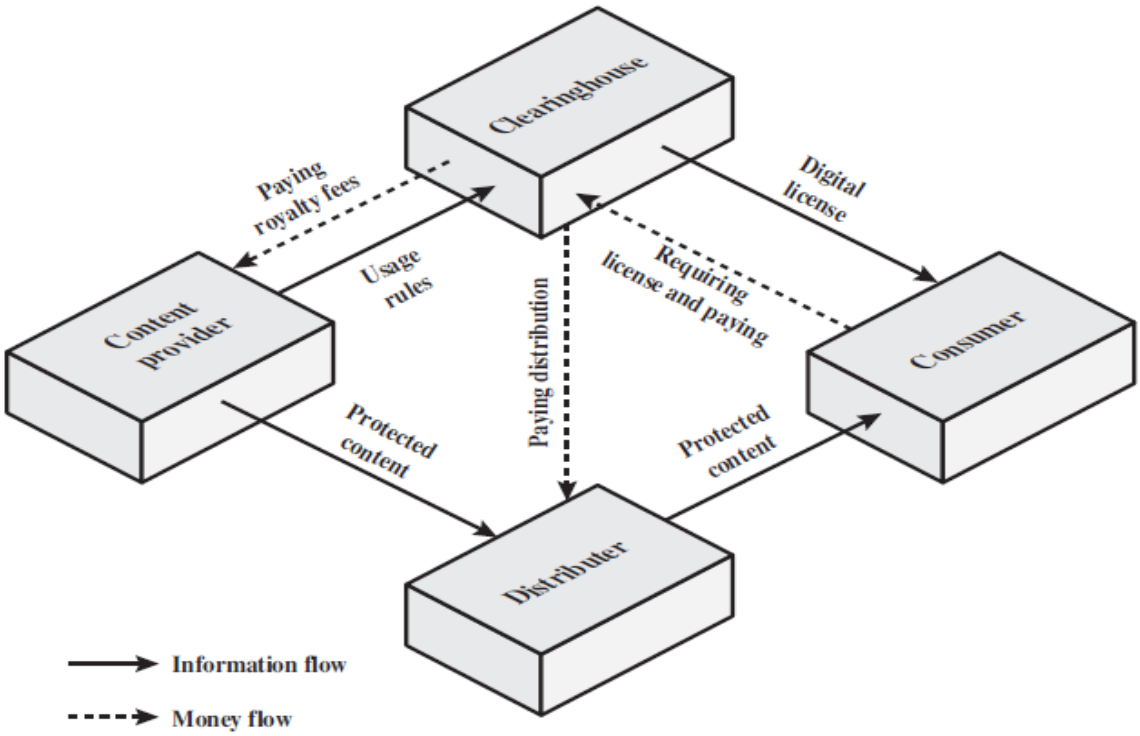
3. Support content use on a variety of platforms, (e.g., PCs, PDAs, iPods, mobile phones).

4. Support content distribution on a variety of media, including CD-ROMs, DVDs, and flash memory.

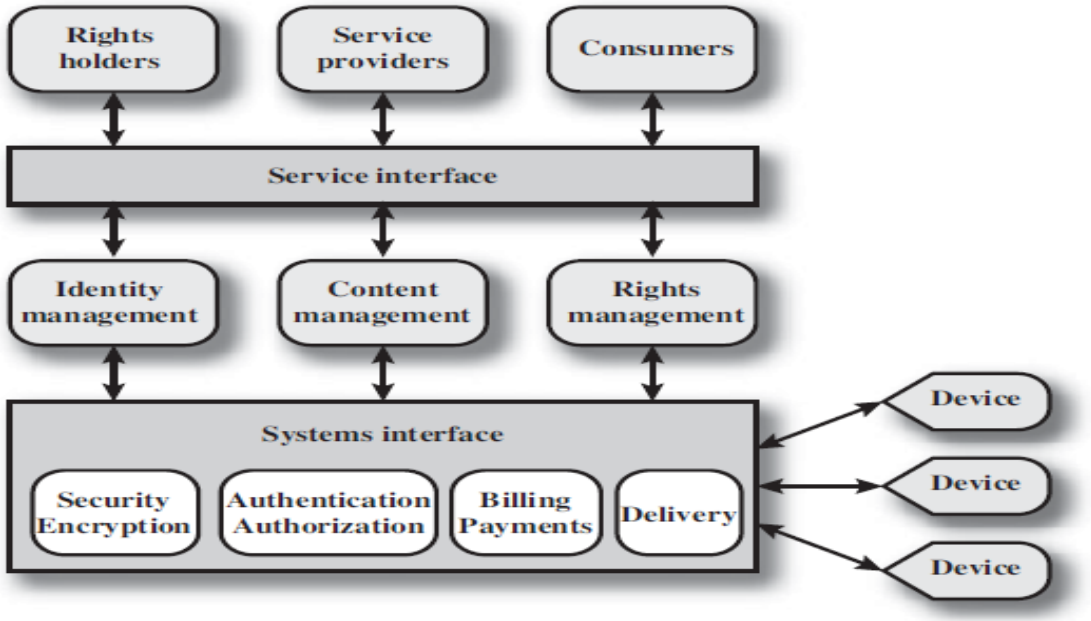
DRM Components:

- Content provider: Holds the digital rights of the content and wants to protect these rights
- Distributor: Provides distribution channels, such as an online shop or a Web retailer.

- Consumer: Uses the system to access the digital content by retrieving downloadable or streaming content through the distribution channel and then paying for the digital license
- Clearinghouse: Handles the financial transaction for issuing the digital license to the consumer and pays royalty fees to the content provider and distribution fees to the distributor accordingly.



DRM System Architecture:



The system is accessed by parties in three roles.

1. Rights holders are the content providers, who either created the content or have acquired rights to the content.

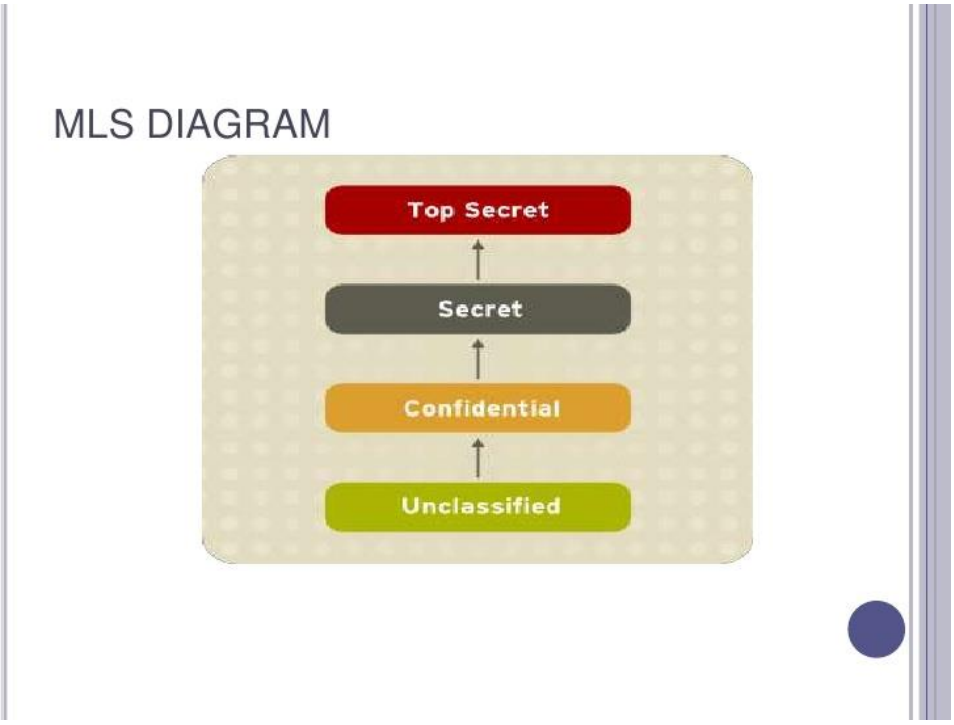
2. Service providers include distributors and clearing houses.
3. Consumers are those who purchase the right to access to content for specific uses

The system interface to the services provided by the DRM system are
Identity management: Mechanisms to uniquely identify entities, such as parties and content
Content management: Processes and functions needed to manage the content lifestyle
Rights management: Processes and functions needed to manage rights, rights holders, and associated requirements

NEED FOR SECURITY AT MULTIPLE LEVELS:

Multilevel security or multiple levels of security (MLS) is the application of a computer system to process information with incompatible classifications (i.e., at different security levels), permit access by users with different security clearances and needs-to-know, and prevent users from obtaining access to information for which they lack authorization.

1)A system that is adequate to protect itself from subversion and has robust mechanisms to separate information domains, that is, **trustworthy**.
2)Refer to an application of a computer that will require the computer to be strong enough to protect itself from subversion and possess adequate mechanisms to separate information domains, that is, a system we must **trust**.



SECURITY POLICIES:

- A **security policy** is a written document in an organization outlining how to protect the organization from threats, including computer **security** threats, and how to handle situations when they do occur.
- A **security policy** must identify all of a company's assets as well as all the potential threats to those assets
- Three main types of policies exist:
    - Organizational (or Master) Policy.
    - System-specific Policy.
    - Issue-specific Policy.

Information security **objectives**

- Integrity—data should be intact, accurate and complete, and IT systems must be kept operational.
- Availability—users should be able to access information or systems when needed.

Structure of a Security Policy:

- Description of the Policy and what is the usage for?
- Where this policy should be applied?
- Functions and responsibilities of the employees that are affected by this policy.
- Procedures that are involved in this policy.
- Consequences if the policy is not compatible with company standards.

1. OSI SECURITY ARCHITECTURE

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements.

ITU-T Recommendation X.800, *Security Architecture for OSI*, defines such a systematic approach. The OSI security architecture is useful to managers as a way of organizing the task of providing security. The OSI security architecture was developed in the context of the OSI protocol architecture.

The OSI security architecture focuses on security attacks, mechanisms, and services:

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

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

Security service: A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to

counter security attacks, and they make use of one or more security mechanisms to provide the service.

Threat

A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Attack

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

S.No	Threats	Attacks
1.	Threat is an attack tends to be an act which is in process	An attack is a threat that has been executed.
2.	Threat can be either intentional or unintentional	Attack is intentional
3.	Threat is a circumstances that has potential to cause loss or damage	Attack is attempted to cause damage
4.	Threat to information system does not mean information was altered or damaged	Attack might cause alteration of information or damage or obtain information.

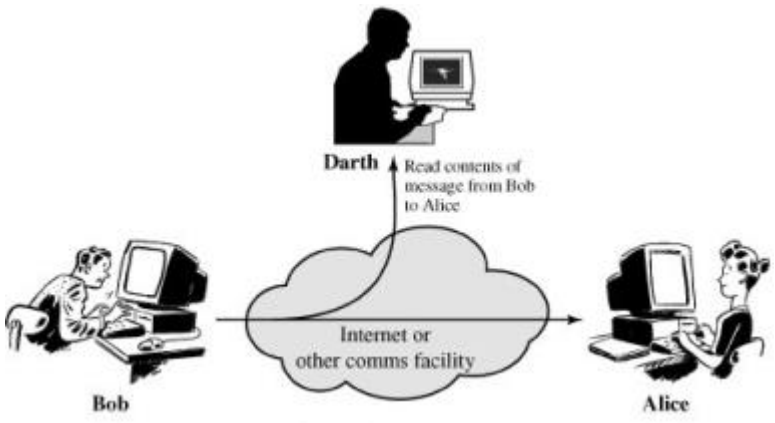
Table 1. Difference between Threats and Attacks

SECURITY ATTACKS:

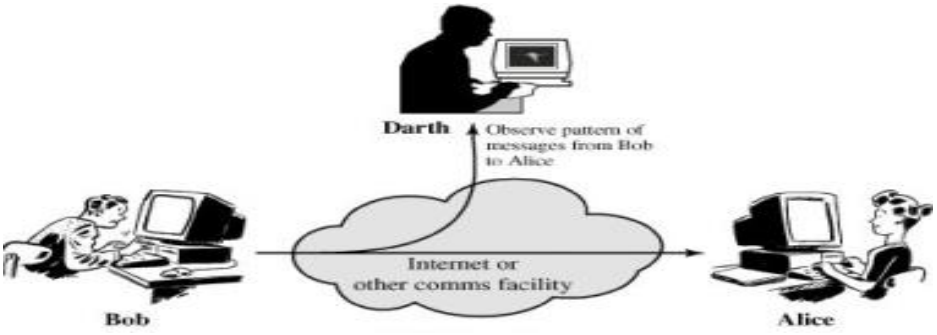
Security attacks could be broadly categorized as

I. Passive attacks: Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. Passive attacks are very difficult to detect because they do not involve any alteration of data. However, it is feasible to prevent the success of these attacks. The goal of the opponent is to obtain information that is being transmitted. Passive attacks are of two types:

Release of message contents: A telephone conversation, an e-mail message and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.

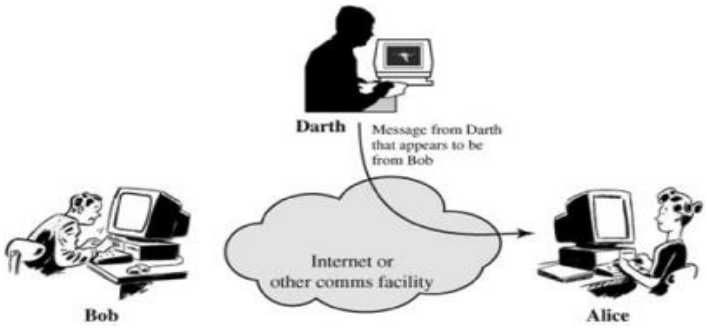


Traffic analysis: If we had encryption protection in place, an opponent might still be able to observe the pattern of the message. The opponent could determine the location and identity of communication hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of communication that was taking place.

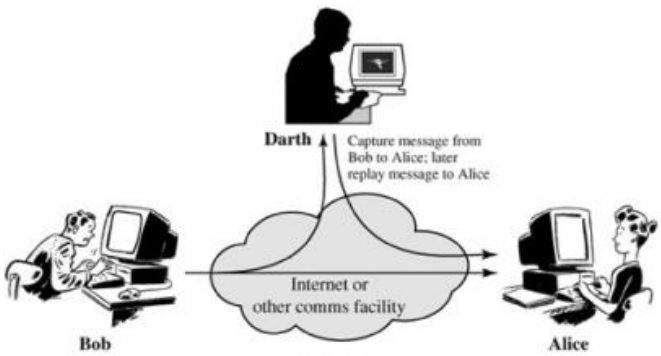


II. Active attacks: These attacks involve some modification of the data stream or the creation of a false stream. It is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communication facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them. These attacks can be classified in to four categories:

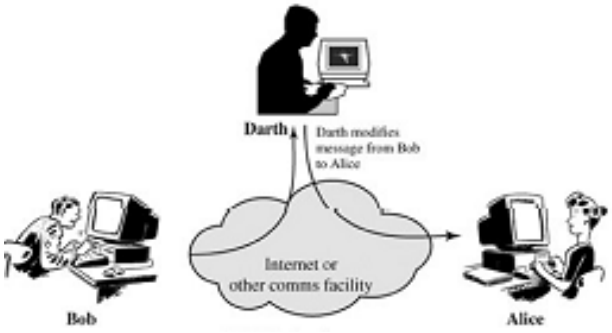
Masquerade (Fabrication)– One entity pretends to be a different entity.



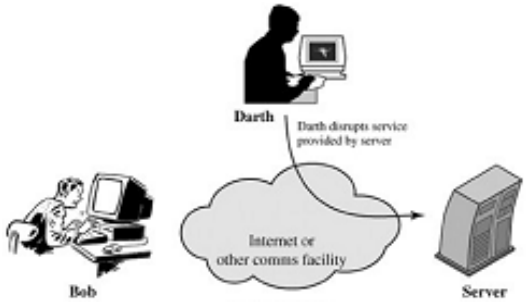
Replay – involves passive capture of a data unit and its subsequent transmission to produce an unauthorized effect.



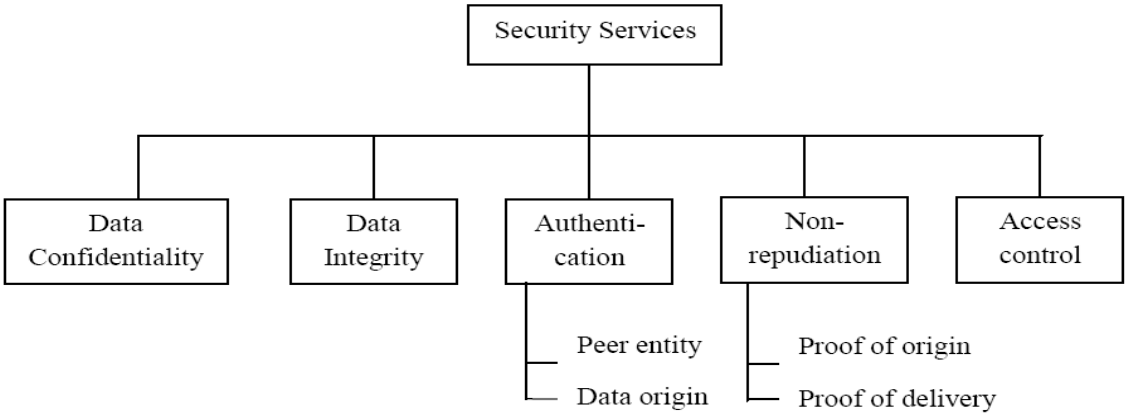
Modification of messages– Some portion of message is altered or the messages are delayed or recorded, to produce an unauthorized effect.



Denial of service – Prevents or inhibits the normal use or management of communication facilities. Another form of service denial is the disruption of an entire network, either by disabling the network or overloading it with messages so as to degrade performance.



SECURITY SERVICES



(i) Authentication: The authentication service is concerned with assuring that a communication is authentic.

Two specific authentication services are defined in X.800:

- **Peer entity authentication:** Provide confidence in the identity of entities connected.
- **Data origin authentication:** Provide assurance that the source of received data is as claimed.

(ii) Access control: Access control is the ability to limit and control the access to host systems and applications.

(iii) Data Confidentiality: Confidentiality is the protection of transmitted data from passive attacks.

- **Connection Confidentiality -** The protection of all user data on a connection
- **Connectionless Confidentiality -** The protection of all user data in a single data block

- **Selective-Field Confidentiality -** The confidentiality of selected fields within the user data on a connection or in a single data block
- **Traffic-Flow Confidentiality -** The protection of the information that might be derived from observation of traffic flows

(iv) Data Integrity: The assurance that data received are exactly as sent by an authorized entity.

- **Connection Integrity with Recovery**
  Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.
- **Connection Integrity without Recovery**
  As above, but provides only detection without recovery.
- **Selective-Field Connection Integrity**
  Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

- **Connectionless Integrity**
  Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.
- **Selective-Field Connectionless Integrity**
  Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

(v) Non repudiation: Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

- **Nonrepudiation, Origin -** Proof that the message was sent by the specified party
- **Nonrepudiation, Destination -** Proof that the message was received by the specified party

SECURITY MECHANISM

- **Encipherment:**
  It uses mathematical algorithm to transform data into a form that is not readily intelligible. It depends upon encryption algorithm and key
- **Digital signature:**
  Data appended to or a cryptographic transformation of a data unit that is to prove integrity of data unit and prevents from forgery
- **Access control**
  A variety of mechanisms that enforce access rights to resources.
- **Data integrity**
  A variety of mechanism are used to ensure integrity of data unit

- **Traffic padding**
  The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
- **Notarization**
  The use of a trusted third party to assure certain properties of a data exchange

SECURITY MECHANISM (X.800 Standard)

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
Encipherment The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
Digital Signature Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**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.
Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.	**Security Recovery** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.
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, especially when a breach of security is suspected.	
Notarization The use of a trusted third party to assure certain properties of a data exchange.	

Table 1.4 Relationship Between Security Services and Mechanisms

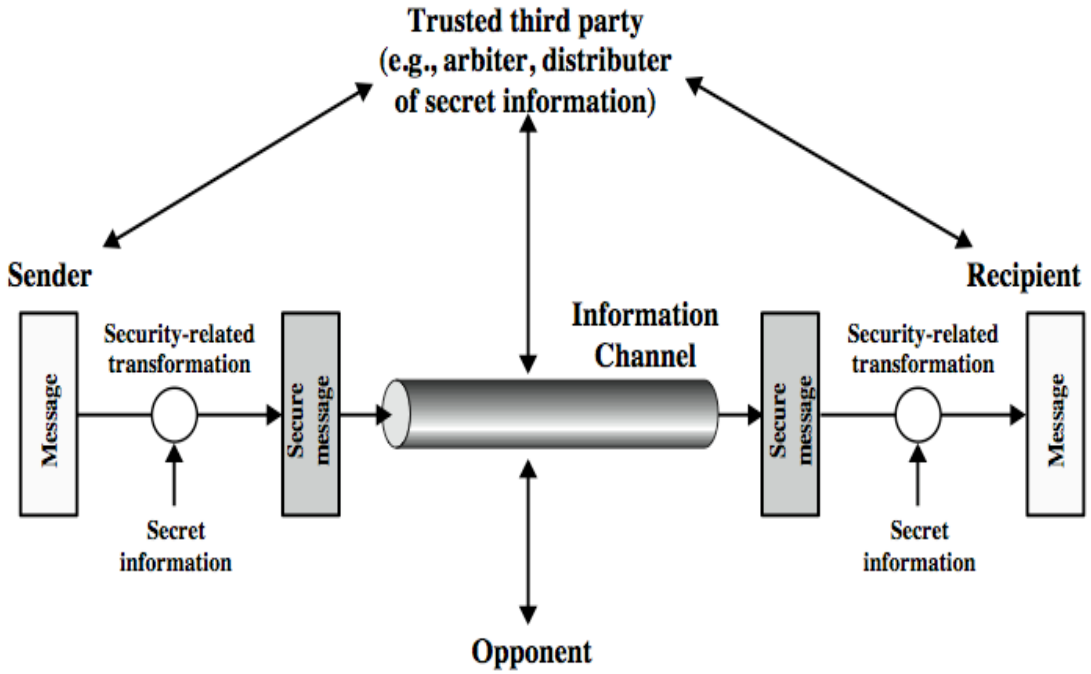
	Mechanism							
Service	**Encipherment**	**Digital Signature**	**Access Control**	**Data Integrity**	**Authentication Exchange**	**Traffic Padding**	**Routing Control**	**Notarization**
Peer Entity Authentication	Y	Y			Y			
Data Origin Authentication	Y	Y						
Access Control			Y					
Confidentiality	Y						Y	
Traffic Flow Confidentiality	Y					Y	Y	
Data Integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

2. A MODEL FOR NETWORK SECURITY

Encryption/Decryption methods fall into two categories.

- Symmetric key
- Public key

In symmetric key algorithms, the encryption and decryption keys are known both to sender and receiver. The encryption key is shared and the decryption key is easily calculated from it. In many cases, the encryption and decryption keys are the same. In public key cryptography, encryption key is made public, but it is computationally infeasible to find the decryption key without the information known to the receiver.



A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, 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 and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

All the techniques for providing security have two components:

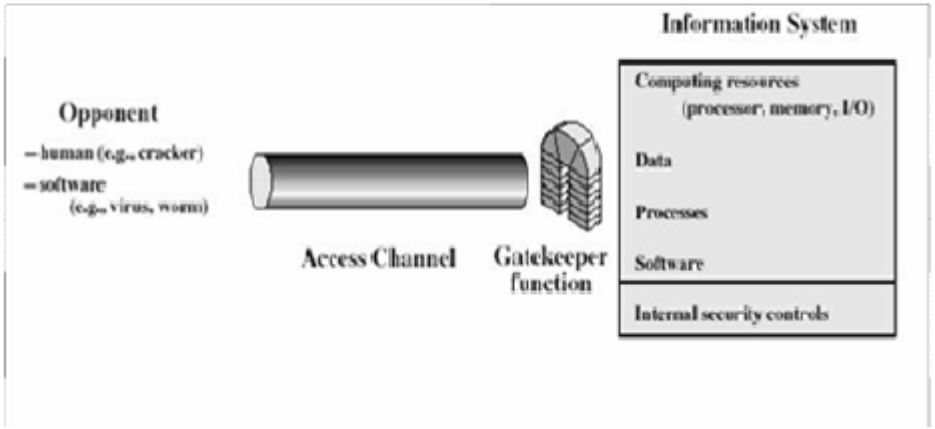
- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent.
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from

any opponent. This general model shows that there are four basic tasks in designing a particular security service:

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

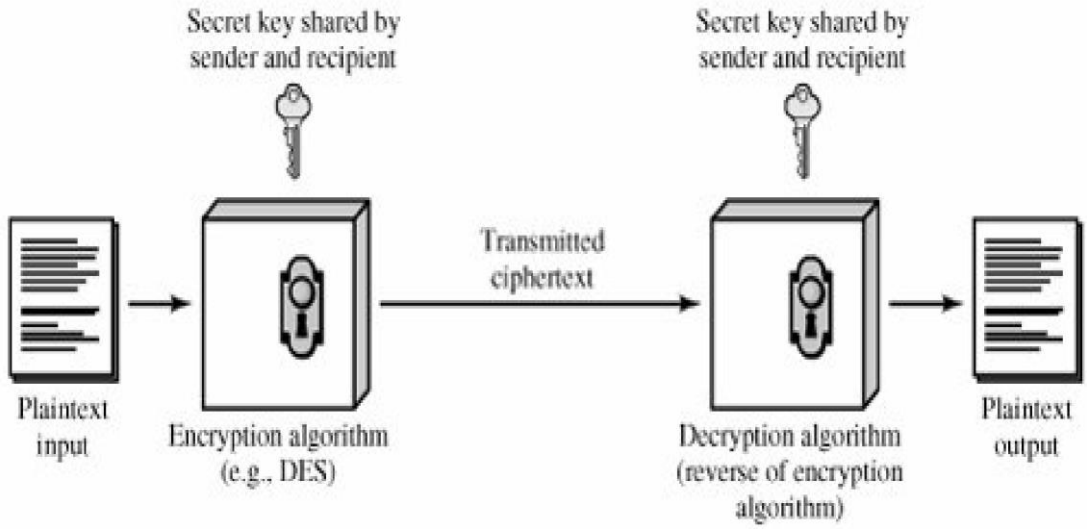
MODEL FOR NETWORK ACCESS SECURITY



Using this model requires us to:
– select appropriate gatekeeper functions to identify users
– implement security controls to ensure only authorized users access designated information or resources
- Trusted computer systems can be used to implement this model

3. CLASSICAL ENCRYPTION TECHNIQUES

Symmetric encryption also referred to as conventional encryption or single-key encryption. Here, the sender and recipient share a common key.



A symmetric encryption scheme has five ingredients

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

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

Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

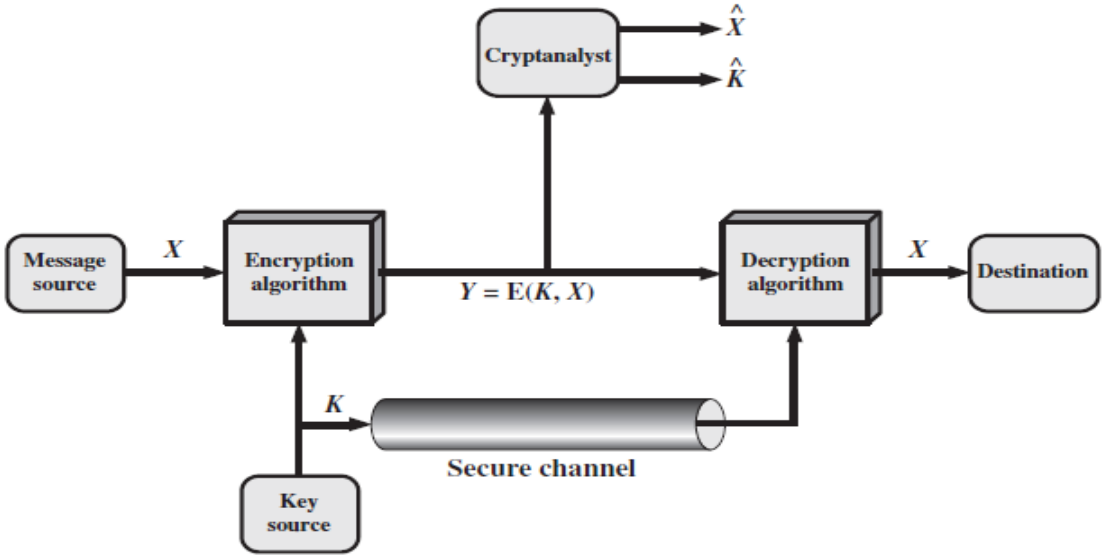
Cipher text: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts. The cipher text is an apparently random stream of data and, as it stands, is unintelligible.

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

There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm.
2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

It is impractical to decrypt a message on the basis of the cipher text *plus* knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret.



Model of symmetric cryptosystem

A source produces a message in plaintext

$X = [X_1, X_2, ..., X_M]$.

M- elements of X are letters.

For encryption, a key of the form

$K = [K_1, K_2, \ldots, K_J]$ is generated.

If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

With the message X and the encryption key K as input, the encryption algorithm forms the cipher text

$Y = [Y_1, Y_2, \ldots, Y_N]$.

$Y = E(K, X)$

Y- cipher text

E- Encryption algorithm

K- Key

X-Plain text

At the receiver side the transformation:

$X = D(K, Y)$

Y- cipher text

D-Decryption algorithm

K- Key

X-Plain text

If the opponent is interested in only this particular message only, tries to find the message estimate $\hat{X}$. But when the opponent is interested in the current and future messages, tries to find key estimate $\hat{K}$.

Cryptographic systems are generally classified along 3 independent dimensions:

 Type of operations used for transforming plain text to cipher text

All the encryption algorithms are based on two general principles:

- **Substitution**, in which each element in the plaintext is mapped into another element
- T**ransposition**, 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 AND BRUTE-FORCE ATTACK

There are two general approaches to attacking a conventional encryption scheme:

- **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm and some knowledge of the general characteristics of the plaintext or even some sample plaintext–cipher text pairs.
- **Brute-force attack:** The attacker tries every possible key on a piece of cipher text until an intelligible translation into plaintext is obtained.

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

Type of Attack	Known to Cryptanalyst
Cipher text Only	• Encryption algorithm • Cipher text
Known Plaintext	• Encryption algorithm • Cipher text • One or more plaintext–cipher text pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Cipher text • Plaintext message chosen by cryptanalyst, together with its corresponding Cipher text generated with the secret key
Chosen Cipher text	• Encryption algorithm • Cipher text • Cipher text chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• Encryption algorithm • Cipher text

	• Plaintext message chosen by cryptanalyst, together with its corresponding Cipher text generated with the secret key • Cipher text chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Encryption algorithms are to be

- **Unconditionally secure**
- **Computationally secure**

An encryption scheme is **unconditionally secure** if the cipher text generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext.

An encryption scheme is said to be **computationally secure**

- If the cost of breaking the cipher exceeds the value of the encrypted information
- If the time required to break the cipher exceeds the useful lifetime of the information.

4. SUBSTITUTION TECHNIQUES

- A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.
- Substitution ciphers can be categorized as either

  i) Monoalphabetic ciphers or ii) polyalphabetic ciphers.
- In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always one-to-one.
- In polyalphabetic substitution, each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the cipher text is one-to-many.

VARIOUS SUBSTITUTION CIPHERS ARE

(i) Caesar Cipher or Shift Cipher Or Additive Cipher (Monoalphabetic Cipher)

(ii) Playfair cipher

(iii) Hill cipher

(iv) Vignere cipher (Poly alphabetic cipher)

(v) Vernam Cipher or One time pad (Poly alphabetic cipher)

(i)CAESAR CIPHER (OR) SHIFT CIPHER

Caeser cipher was proposed by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.

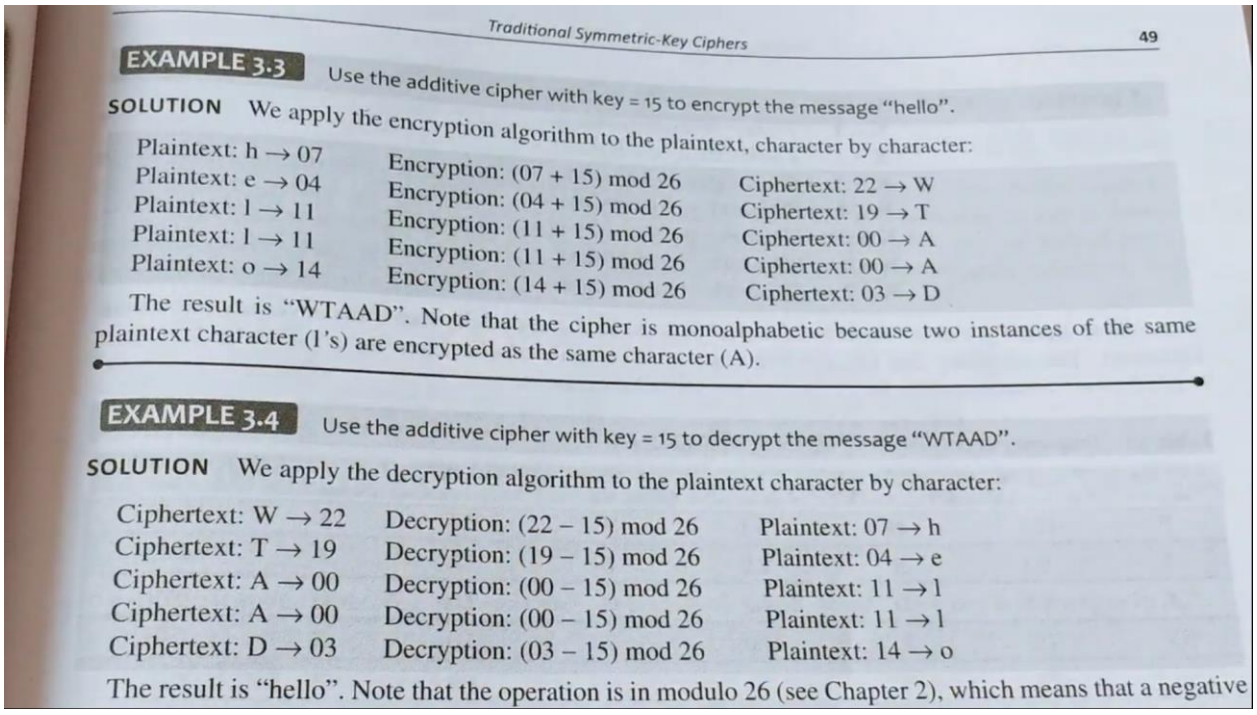
```
plain:  meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

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

Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25



Traditional Symmetric-Key Ciphers 49

EXAMPLE 3.3 Use the additive cipher with key = 15 to encrypt the message "hello".

SOLUTION We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

The result is "WTAAD". Note that the cipher is monoalphabetic because two instances of the same plaintext character (l's) are encrypted as the same character (A).

EXAMPLE 3.4 Use the additive cipher with key = 15 to decrypt the message "WTAAD".

SOLUTION We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W → 22	Decryption: (22 – 15) mod 26	Plaintext: 07 → h
Ciphertext: T → 19	Decryption: (19 – 15) mod 26	Plaintext: 04 → e
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: D → 03	Decryption: (03 – 15) mod 26	Plaintext: 14 → o

The result is "hello". Note that the operation is in modulo 26 (see Chapter 2), which means that a negative

Note that the alphabet is wrapped around, so that letter following 'z' is 'a'.

For each plaintext letter p, substitute the cipher text letter c such that

C = **E(3, P = (P+3) mod 26**

Decryption is

P=D(3,c)=(C-3) mod 26

The general Caesar algorithm is

C = E(K, P) = (P + K) mod 26

where *k* takes on a value in the range 1 to 25.

The decryption algorithm is simply

P = D(K, C) = (C - K) mod 26

If it is known that a given cipher text is a Caesar cipher, then a brute-force cryptanalysis is easily performed: simply try all the 26 possible keys.

Cryptanalysis of Caesar Cipher

1. The encryption and decryption algorithms are known
2. There are only 26 possible keys. Hence brute force attack takes place
3. The language of the plaintext is known and easily recognizable

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Brute-Force Cryptanalysis of Caesar Cipher

(ii) MONOALPHABETIC CIPHER

- Each plaintext letter maps to a different random cipher text letter
- Here, 26! Possible keys are used to eliminate brute force attack

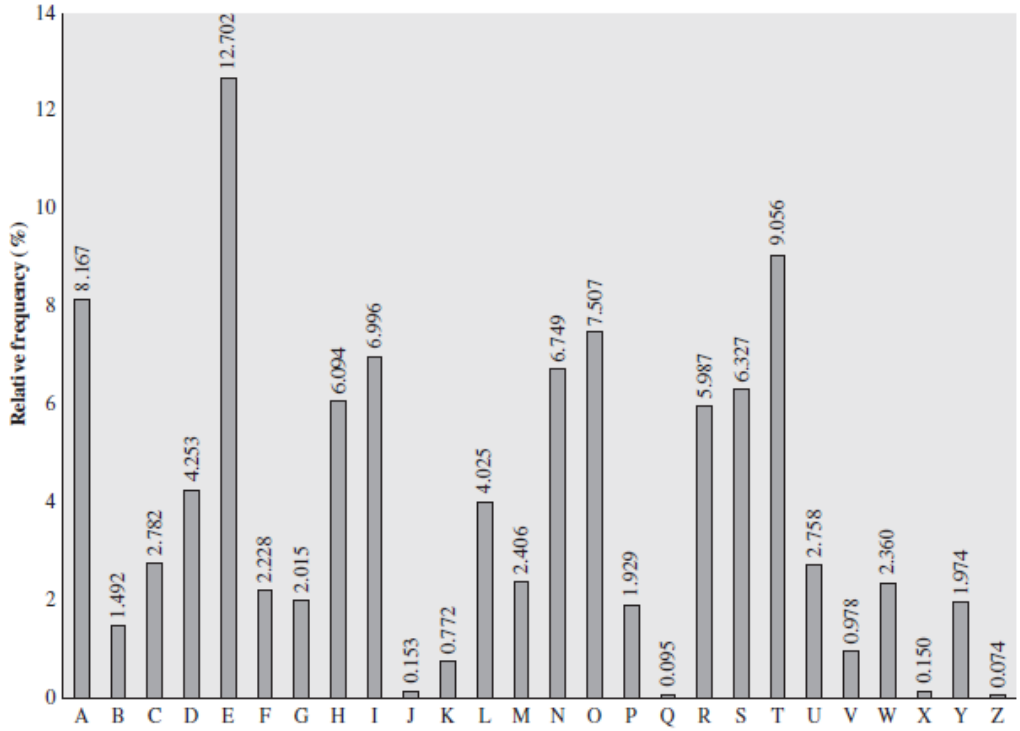
There is, however, another line of attack. If the cryptanalyst knows the nature of the plaintext (e.g., non-compressed English text), then the analyst can exploit the regularities of the language.

MONOALPHABETIC CIPHER:

- A Single Cipher alphabets for each plain text alphabet is used throughout the process.
- Rather than just shifting the alphabet could shuffle (jumble) the letters arbitrarily
- Relationship between a character in the plain text to a symbol in cipher text is always one to one
- Assume the below table(without the repetition of letter, any letters can be mapped)
  Plain: a b c de f g h I j k l m n o p q r s t u v w x y z
  Cipher: DKVQF IB JWPESCXHTMYAUOLRGZN
- **Example 1(By referring the above table)**
  Plain text : MY NAME
  Cipher Text: CZ XDCF
- From the above example, we say that wherever we use M. The M letter can replaced by C. (i.e) A Single Cipher alphabets for each plain text alphabet is used throughout the process.

As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				



Relative frequency of letters in English text

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
 t a        e  e te  a that e e a       a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
   e t   ta t ha e ee  a e  th    t  a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
 e  e e tat  e   the   t
```

Only four letters have been identified, but already we have quite a bit of the message. Continued analysis of frequencies plus trial and error should easily yield a solution from this point. The complete plaintext, with spaces added between words, follows:

```
it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow
```

(iii) PLAYFAIR CIPHER

The best known multiple letter encryption cipher is the playfair, which treats diagrams in the plaintext as single units and translates these units into cipher text diagrams. The playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword.

Let the keyword be "monarchy".

The matrix is constructed by

- Filling in the letters of the keyword from left to right and from top to bottom
- Duplicates are removed
- Remaining unfilled cells of the matrix is filled with remaining alphabets in alphabetical order.

The matrix is 5x5. It can accommodate 25 alphabets. To accommodate the 26th alphabet I and J are counted as one character.

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

Rules for encryption

- Repeating plaintext letters that would fall in the same pair are separated with a filler letter such as 'x'.
- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.

- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

Example

Plain text: Balloon

Ba ll oo n

Ba lx lo on

Ba→I/JB

lx→SU

lo→PM

on→NA

Strength of playfair cipher

- Playfair cipher is a great advance over simple mono alphabetic ciphers.
- Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual digram is more difficult.
- Frequency analysis is much more difficult.

Disadvantage

Easy to break because it has the structure and the resemblance of the plain text language

(iv) HILL CIPHER

It is a multi-letter cipher. It is developed by Lester Hill. The encryption algorithm takes m successive plaintext letters and substitutes for them m cipher text letters. The substitution is determined by m linear equations in which each character is assigned numerical value (a=0,b=1…z=25). For m =3 the system can be described as follows:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} K_{11}K_{12}K_{13} \\ K_{21}K_{22}K_{23} \\ K_{31}K_{32}K_{33} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} \bmod 26$$

C=KP mod 26

C and P are column vectors of length 3 representing the cipher and plain text respectively.
Consider the message 'ACT', and

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

The key below (or GYBNQKURP in letters)

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

Thus the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26}$$

which corresponds to a ciphertext of 'POH'

Decryption

Decryption algorithm is done as **$P=K^{-1}C \bmod 26$**

In order to decrypt, we turn the ciphertext back into a vector, then simply multiply by the inverse matrix of the key matrix (IFKVIVVMI in letters).

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$$

Cipher text of 'POH'

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \equiv \begin{pmatrix} 260 \\ 574 \\ 539 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} \pmod{26}$$

Now gets us back the plain text 'ACT'

Merits and Demerits

- Completely hides single letter and 2 letter frequency information.
- Easily attacked with known plain text attack

(v)POLYALPHABETIC CIPHERS

Poly alphabetic cipher is a simple technique to improve mono-alphabetic technique.

- There is no fixed substitution
- Each Occurrence of a character may have a different substitute (i.e) we can use more than one substitution for the same letter

Plain Text : MY NAME
Cipher Text: NP OBXZ (here, M Letter can be replaced with N and X. No fixed substitute)

The features are

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

Example: **Vigenere Cipher**

Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of encryption is simple: Given a key letter x and a plaintext letter y, the cipher text is at the intersection of the row labelled x and the column labelled y; in this case, the cipher text is V. To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword.

Key=deceptive

Plain text= we are discovered save yourself

e.g., key = d e c e p t i v e d e c e p t i v e d e c e p t i v e

PT = w e a r e d i s c o v e r e d s a v e y o u r s e l f

CT = ZICVTWQNGRZGVTWAVZHCQYGLMGJ

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

	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

EXAMPLE 2 :

```
key:            deceptivedeceptivedeceptive
plaintext:      wearediscoveredsaveyourself
ciphertext:     ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

Expressed numerically, we have the following result.

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9

Strength of Vigenere cipher

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

(vi) VERNAM CIPHER or ONE-TIME PAD

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

The system can be expressed as follows:

$$C_i = P_i \oplus K_i$$

C_i - ith binary digit of cipher text P_i - i^{th} binary digit of plaintext K_i - ith binary digit of key

$\oplus$ – exclusive OR operation

Thus the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key. Because of the properties of XOR, decryption simply involves the same bitwise operation:

$$P_i = C_i \oplus K_i$$

e.g., plaintext = 0 0 1 0 1 0 0 1
Key = 1 0 1 0 1 1 0 0
ciphertext = 1 0 0 0 0 1 0 1

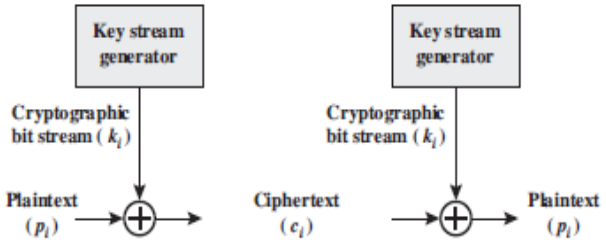


Figure 2.7 Vernam Cipher

Advantages

- It is unbreakable since cipher text bears no statistical relationship to the plaintext
- Not easy to break

Drawbacks

- Practically impossible to generate a random key as to the length of the message
- The second problem is that of key distribution and key protection.

Due to the above two drawbacks, one time pad is of limited use and is used for low band width channel which needs high security.

5. TRANSPOSITION TECHNIQUES

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.

i) **Rail Fence Cipher**

It 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 Cipher text MEATECOLOSETTHSHOHUE

ii) Row Transposition Ciphers-

A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of columns then becomes the key of the algorithm.

e.g., plaintext = meet at the school house

Key = 4 3 1 2 5 6 7

PT = m e e t a t t

h e s c h o o

l h o u s e x

CT = ESOTCUEEHMHLAHSTOETOX

Demerits

- Easily recognized because the frequency is same in both plain text and cipher text.
- Can be made secure by performing more number of transpositions.

6. STEGANOGRAPHY

In Steganography, the plaintext is hidden. The existence of the message is concealed. For example, the sequence of first letters of each word of the overall message spells out the hidden message.

Various other techniques have been used historically; some examples are the following:

• **Character marking:** Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at 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 up in front of a light.
• **Typewriter correction ribbon:** Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawback

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

MODERN CRYPTOGRAPHY:

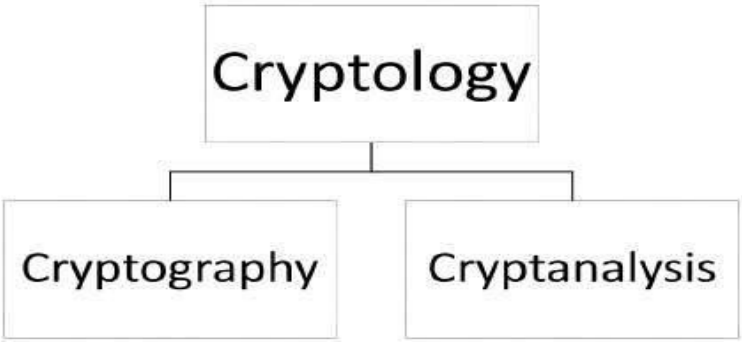
Modern cryptography is the cornerstone of computer and communications security. Its foundation is based on various concepts of mathematics such as number theory, computational-complexity theory, and probability theory.

Difference between Classical Cryptography and Modern Cryptography:

Classic Cryptography	Modern Cryptography
It manipulates traditional characters, i.e., letters and digits directly.	It operates on binary bit sequences.
It is mainly based on 'security through obscurity'.	It relies on publicly known mathematical algorithms for coding the information.
It requires the entire cryptosystem for communicating confidentially.	Modern cryptography requires parties interested in secure communication to possess the secret key only.

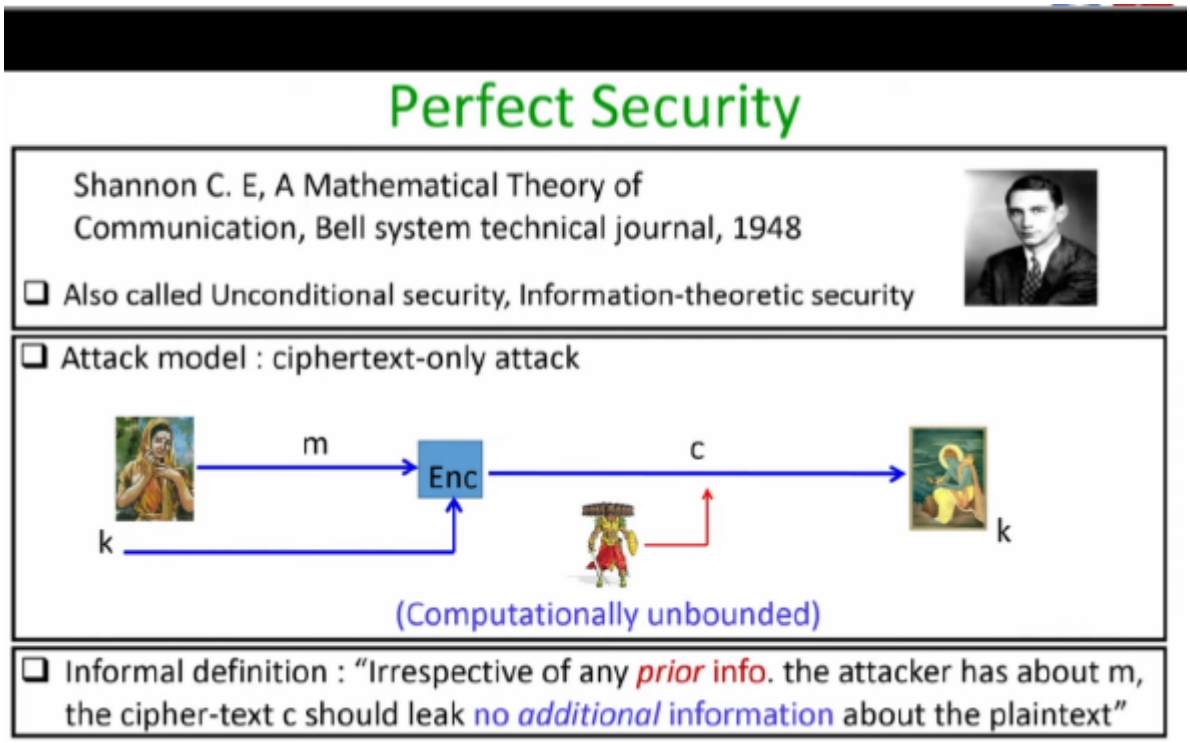
Cryptology:

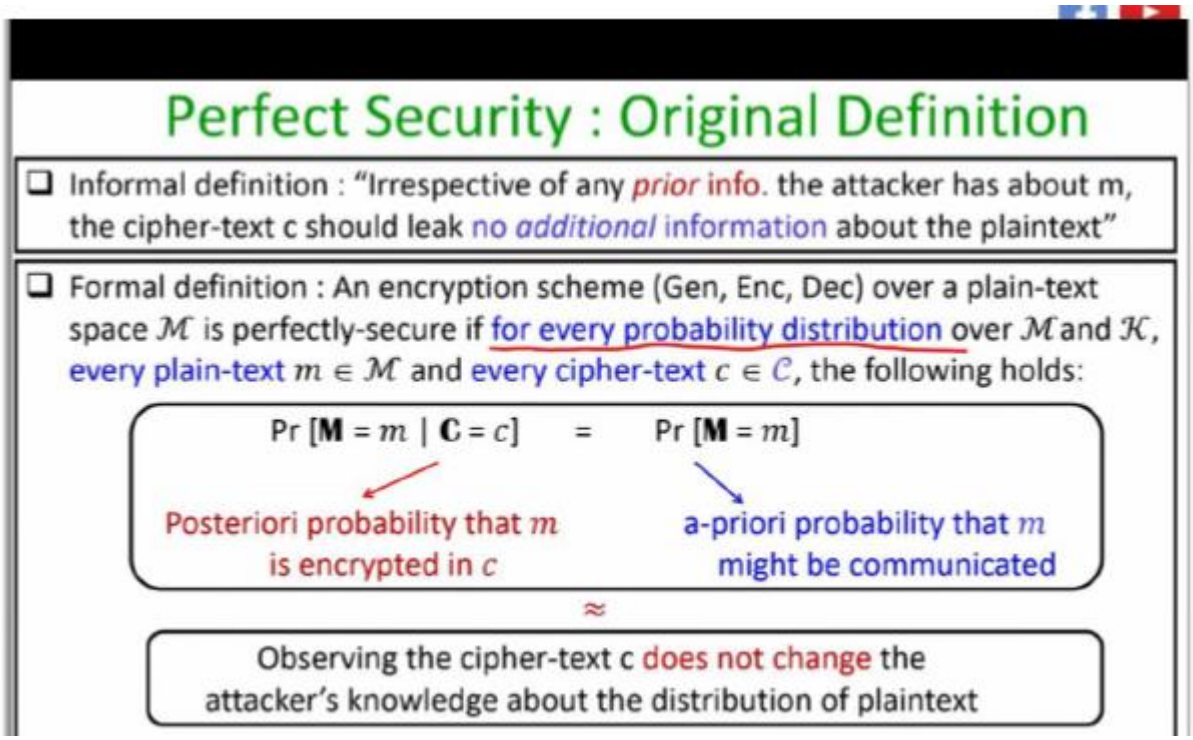
- Cryptography is the art and science of making a cryptosystem that is capable of providing information security.
- The art and science of breaking the cipher text is known as Cryptanalysis
- Cryptology is the study of codes, both creating and solving them.

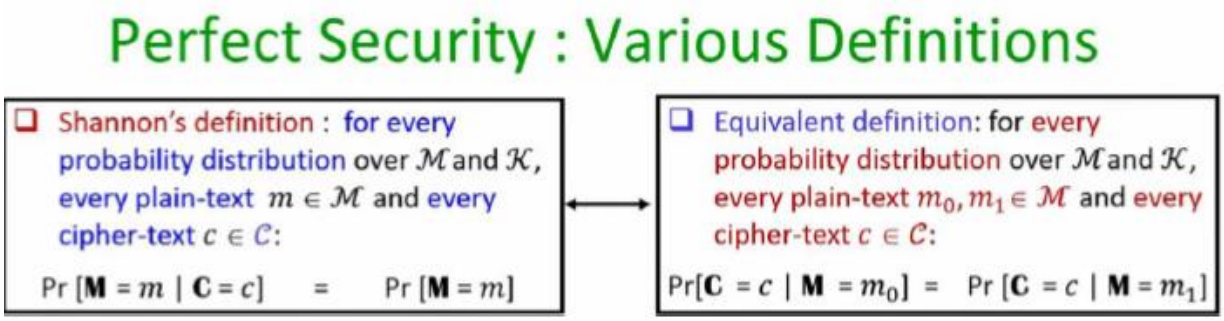


Perfect Security:

Perfect Secrecy (or Information-theoretic secure) means that the ciphertext conveys no information about the content of the plaintext. In effect this means that, no matter how much ciphertext you have, it does not convey anything about what the plaintext and key were.







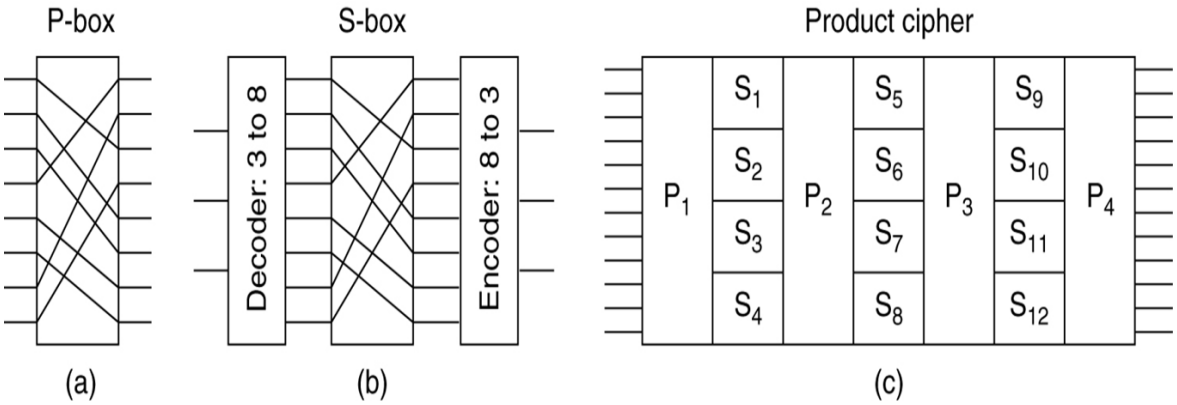
Interpretation:
1)Probability of knowing a plain text remains the same before and after seeing the cipher text
2)Probability distribution of cipher text is independent of Plain text.
Perfect secrecy has nothing to do with plaintext distribution. Thus a crypto-scheme will achieve perfect secrecy irrespective of the language used in the plaintext.

Information theory:

- Concepts, methods and results from coding theory and information theory are widely used in cryptography and cryptanalysis. See the article ban (unit) for a historical application.
- Information theory is also used in information retrieval, intelligence gathering, gambling, statistics, and even in musical composition.
- A key measure in information theory is Entropy. Entropy quantifies the amount of uncertainty involved in the value of a random variable or the outcome of a random process.

- In terms of Cryptography, entropy must be supplied by the cipher for injection into the plaintext of a message so as to neutralise the amount of structure that is present in the unsecure plaintext message.

Product Cryptosystem:



Basic elements of product ciphers.
(a) P-box. (b) S-box. (c) Product.

Cryptanalysis:

Cryptanalysis is the study of ciphertext, ciphers and cryptosystems with the aim of understanding how they work and finding and improving techniques for defeating or weakening them.

UNIT II SYMMETRIC KEY CRYPTOGRAPHY

MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY: Algebraic structures - Modular arithmetic-Euclid‟s algorithm- Congruence and matrices - Groups, Rings, Fields- Finite fields

SYMMETRIC KEY CIPHERS: SDES – Block cipher Principles of DES – Strength of DES – Differential and linear cryptanalysis - Block cipher design principles – Block cipher mode of operation – Evaluation criteria for AES – Advanced Encryption Standard - RC4 – Key distribution.

ALGEBRAIC STRUCTURES

- Cryptography requires sets of integers and specific operations that are defined for those sets.
- The combination of the set and the operations that are applied to the elements of the set is called an Algebraic Structure.

MODULAR ARITHMETIC

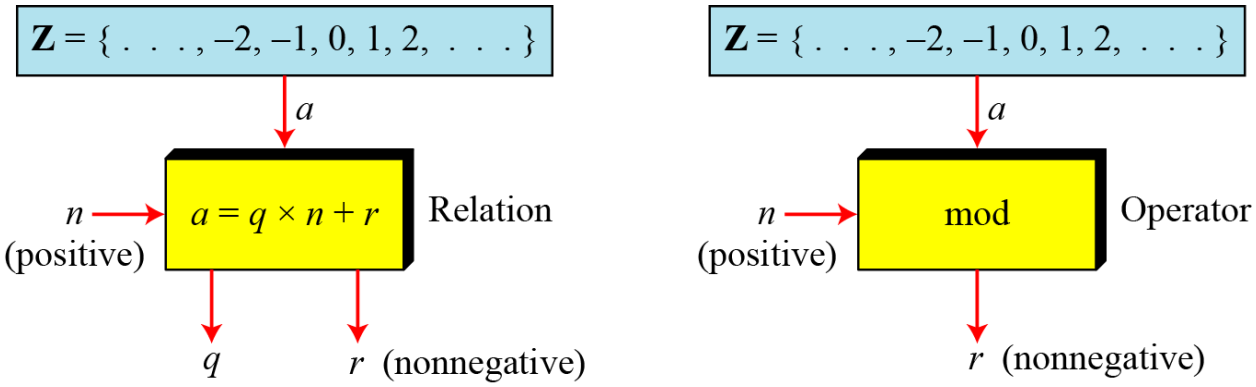
If *a* is an integer and *n* is a positive integer, we define *a* mod *n* to be the remainder when *a* is divided by *n*. The integer *n* is called the **modulus**.

$$a = qn + r \qquad 0 \le r < n;$$

$$q = \lfloor a/n \rfloor$$

The division relationship (a = q × n + r) discussed in the previous section has two inputs (a and n) and two outputs (q and r). In modular arithmetic, we are interested in only one of the outputs, the remainder r.

The modulo operator is shown as mod. The second input (n) is called the modulus. The output r is called the residue.



Examples

Find the result of the following operations:

a. 27 mod 5 **b. 36 mod 12**

Solution

a. Dividing 27 by 5 results in r = 2
b. Dividing 36 by 12 results in r = 0.

CONGRUENCE

Two integers a and b are said to be congruent modulo n if

a (mod n)≡ b (mod n)

a ≡ b (mod n)

73 ≡ 4 mod 23

Properties of Congruences

Congruences have the following properties:

1. a≡ b (mod n) if n|(a-b)
2. a≡ b (mod n) implies b≡ a(mod n)
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

To show that two integers are congruent, we use the congruence operator (≡). For example, we write:

$$2 \equiv 12 \pmod{10} \qquad 13 \equiv 23 \pmod{10}$$
$$3 \equiv 8 \pmod{5} \qquad 8 \equiv 13 \pmod{5}$$

Example:

Perform the following operations (the inputs come from Zn):

a. Add 7 to 14 in Z15.

b. Subtract 11 from 7 in Z13.

c. Multiply 11 by 7 in Z20.

Solution

$$(14 + 7) \bmod 15 \rightarrow (21) \bmod 15 = 6$$
$$(7 - 11) \bmod 13 \rightarrow (-4) \bmod 13 = 9$$
$$(7 \times 11) \bmod 20 \rightarrow (77) \bmod 20 = 17$$

MODULAR ARITHMETIC OPERATIONS

Modular arithmetic exhibits the following properties:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$

2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$

3. $[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$

Example: 11 mod 8 = 3; 15 mod 8 = 7
[(11 mod 8) + (15 mod 8)] mod 8 = 10 mod 8 = 2
(11 + 15) mod 8 = 26 mod 8 = 2
[(11 mod 8) - (15 mod 8)] mod 8 = -4 mod 8 = 4
(11 - 15) mod 8 = -4 mod 8 = 4
[(11 mod 8) * (15 mod 8)] mod 8 = 21 mod 8 = 5
(11 * 15) mod 8 = 165 mod 8 = 5

RELATIVELY PRIME

Two integers are **relatively prime,** if their only common positive integer factor is 1.
8 and 15 are relatively prime because
Positive divisors of 8 are 1,2,4,8
Positive divisors of 15 are 1, 3, 5, 15
Therefore, common positive factor=1.

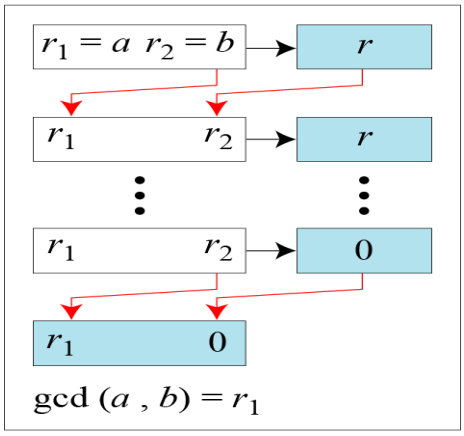
EUCLIDEAN ALGORITHM

Euclidean algorithm is a simple procedure for determining the greatest common divisor of two positive integers.

The positive integer *c* is said to be the greatest common divisor of *a* and *b* if

1. *c* is a divisor of *a* and of *b*.
2. Any divisor of *a* and *b* is a divisor of *c*.

Fact 1: gcd (a, 0) = a
Fact 2: gcd (a, b) = gcd (b, r), where r is the remainder of dividing a by b



a. Process

$r_1 \leftarrow a;$ $r_2 \leftarrow b;$ (Initialization)
while $(r_2 > 0)$
{
$q \leftarrow r_1 / r_2;$
$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2;$ $r_2 \leftarrow r;$
}
gcd $(a, b) \leftarrow r_1$

b. Algorithm

EUCLID(*a*, *b*)
1. A← *a*; B←*b*
2. if B = 0 **return** A = gcd(*a*, *b*)
3. R = A mod B
4. A←B
5. B←R
6. goto 2

Euclidean Algorithm Revisited

For any integers *a, b,* with $a \geq b \geq 0$,
gcd(*a, b*) = gcd(*b, a* mod *b*)

Example 1

gcd(55, 22) = gcd(22, 55 mod 22) = gcd(22, 11) = 11
gcd(18, 12) = gcd(12, 6) = gcd(6, 0) = 6
gcd(11, 10) = gcd(10, 1) = gcd(1, 0) = 1

Example 2 Find the greatest common divisor of 2740 and 1760.

Solution: We have gcd (2740, 1760) = 20.

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

Example 3 Find the greatest common divisor of 25 and 60.

Solution : We have gcd (25, 65) = 5.

q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

Euclidean Algorithm	
Calculate	**Which satisfies**
$r_1 = a \bmod b$	$a = q_1 b + r_1$
$r_2 = b \bmod r_1$	$b = q_2 r_1 + r_2$
$r_3 = r_1 \bmod r_2$	$r_1 = q_3 r_2 + r_3$
•	•
•	•
•	•
$r_n = r_{n-2} \bmod r_{n-1}$	$r_{n-2} = q_n r_{n-1} + r_n$
$r_{n+1} = r_{n-1} \bmod r_n = 0$	$r_{n-1} = q_{n+1} r_n + 0$ $d = \gcd(a, b) = r_n$

Recursive function:

If (b=0) then return a;

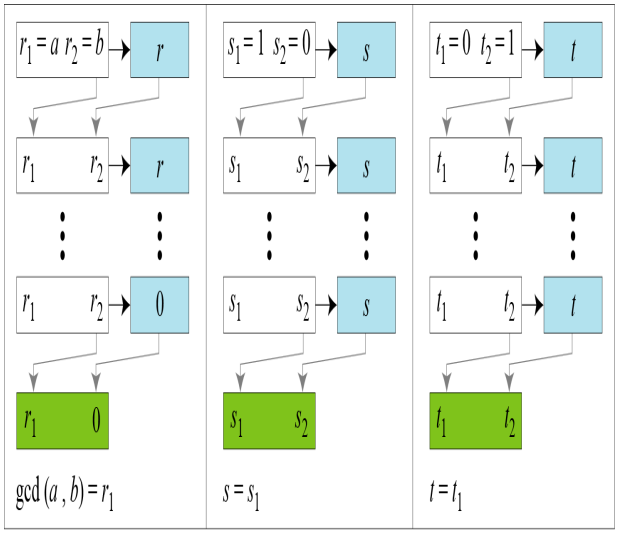
else return Euclid(b, a mod b);

EXTENDED EUCLIDEAN ALGORITHM

Given two integers *a* and *b*, we often need to find other two integers, *s* and *t*, such that

$$s \times a + t \times b = \gcd(a, b)$$

The extended Euclidean algorithm can calculate the gcd (*a*, *b*) and at the same time calculate the value of *s* and *t*.



a. Process

$r_1 \leftarrow a;$ $r_2 \leftarrow b;$
$s_1 \leftarrow 1;$ $s_2 \leftarrow 0;$
$t_1 \leftarrow 0;$ $t_2 \leftarrow 1;$ (Initialization)

while $(r_2 > 0)$

{

$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2;$ $r_2 \leftarrow r;$ (Updating *r*'s)

$s \leftarrow s_1 - q \times s_2;$
$s_1 \leftarrow s_2;$ $s_2 \leftarrow s;$ (Updating *s*'s)

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2;$ $t_2 \leftarrow t;$ (Updating *t*'s)

}

gcd $(a , b) \leftarrow r_1;$ $s \leftarrow s_1;$ $t \leftarrow t_1$

b. Algorithm

Example1: Given *a* = 161 and *b* = 28, find gcd (*a*, *b*) and the values of *s* and *t*.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	−5
1	28	21	7	0	1	−1	1	−5	6
3	21	7	0	1	−1	4	−5	6	−23
	7	0		**−1**	4		**6**	−23	

Solution

We get gcd (161, 28) = 7, $s = -1$ and $t = 6$.

POLYNOMIAL ARITHMETIC

A **polynomial** of degree n (integer $n \geq 0$) is an expression of the form

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

where the a_i – *coefficients*

$$f(x) = \sum_{i=0}^{n} a_i x^i; \quad g(x) = \sum_{i=0}^{m} b_i x^i; \quad n \geq m$$

Addition is defined as

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i)x^i + \sum_{i=m+1}^{n} a_i x^i$$

Multiplication is defined as

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

where

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0$$

Eg.: Let $f(x) = x^3 + x^2 + 2$ and $g(x) = x^2 - x + 1$, where S is the set of integers. Then
$f(x) + g(x) = x^3 + 2x^2 - x + 3$
$f(x) - g(x) = x^3 + x + 1$
$f(x) * g(x) = x^5 + 3x^2 - 2x + 2$

Example 1 Find gcd[$a(x)$, $b(x)$] for $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ and $b(x) = x^4 + x^2 + x + 1$.

Euclidean algorithm to compute the greatest common divisor of two polynomials

gcd[$a(x)$, $b(x)$] = gcd[$b(x)$, $a(x)$ mod $b(x)$]

=gcd(b(x),r1(x))

=gcd[r1(x),b(x) mod r1(x)]

$$
\begin{array}{r}
x^2 + x \\
x^4 + x^2 + x + 1 \overline{)\, x^6 + x^5 + x^4 + x^3 + x^2 + x + 1} \\
\underline{x^6 \quad + x^4 + x^3 + x^2} \\
x^5 \qquad\qquad + x + 1 \\
\underline{x^5 \qquad x^3 + x^2 + x} \\
x^3 + x^2 \qquad + 1
\end{array}
$$

This yields $r_1(x) = x^3 + x^2 + 1$ and $q_1(x) = x^2 + x$.
Then, we divide $b(x)$ by $r_1(x)$.

$$
\begin{array}{r}
x + 1 \\
x^3 + x^2 + 1 \overline{)\, x^4 \quad + x^2 + x + 1} \\
\underline{x^4 + x^3 \quad + x} \\
x^3 + x^2 \quad + 1 \\
\underline{x^3 + x^2 \quad + 1}
\end{array}
$$

This yields $r_2(x) = 0$ and $q_2(x) = x + 1$.
Therefore, $\gcd[a(x), b(x)] = r_1(x) = x^3 + x^2 + 1$.

MULTIPLICATIVE INVERSE

It is easy to find the multiplicative inverse of an element in GF(p) for small values of p *by* constructing a multiplication table, such as shown in Table and the desired result can be read directly. However, for large values of p, this approach is not practical.

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) Addition modulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

w	$-w$	w^{-1}
0	0	–
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) Additive and multiplicative inverses modulo 7

If a and b are relatively prime, then b has a multiplicative inverse modulo a. That is, if gcd(a, b) = 1, then b has a multiplicative inverse modulo a. That is, for positive integer $b < a$, there exists a b-1 < a such that

$bb^{-1} = 1 \bmod a$.

If a is a prime number and $b < a$, then clearly a and b are relatively prime and have a greatest common divisor of 1. We now show that we can easily compute b-1 using the extended Euclidean algorithm.

Finding the Multiplicative Inverse of a polynomial

Polynomial Arithmetic Modulo $(x^3 + x + 1)$

	+	000	001	010	011	100	101	110	111
		0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+1	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(a) Addition

	×	000	001	010	011	100	101	110	111
		0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+1	x^2	$x+1$

(b) Multiplication

CONGRUENCE AND MATRICES

For a positive integer n, two integers a and b are said to be congruent modulo n (or a is congruent to b modulo n), if a and b have the same remainder when divided by n (or equivalently if a − b is divisible by n). It can be expressed as a ≡ b mod n.

Matrices: ***A matrix of size l X m***

$$\text{Matrix } \mathbf{A}: \quad \overset{\textit{l} \text{ rows}}{\overbrace{\phantom{\begin{matrix}a\\a\\a\\a\end{matrix}}}} \quad \overset{\textit{m} \text{ columns}}{\overbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix}}}$$

Examples of matrices

$$\begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} \quad \begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Row matrix | Column matrix | Square matrix | **0** | **I**

Operations and Relations: **Addition and subtraction of matrices**

$$\begin{bmatrix} 12 & 4 & 4 \\ 11 & 12 & 30 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} + \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

$$\mathbf{C = A + B}$$

$$\begin{bmatrix} -2 & 0 & -2 \\ -5 & -8 & 10 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} - \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

$$\mathbf{D = A - B}$$

Multiplication of a row matrix by a column matrix:

$$\underset{\mathbf{C}}{\begin{bmatrix} 53 \end{bmatrix}} = \underset{\mathbf{A}}{\begin{bmatrix} 5 & 2 & 1 \end{bmatrix}} \times \underset{\mathbf{B}}{\begin{bmatrix} 7 \\ 8 \\ 2 \end{bmatrix}}$$

In which: $53 = 5 \times 7 + 2 \times 8 + 1 \times 2$

Multiplication of a 2 × 3 matrix by a 3 × 4 matrix

$$\underset{\mathbf{C}}{\begin{bmatrix} 52 & 18 & 14 & 9 \\ 41 & 21 & 22 & 7 \end{bmatrix}} = \underset{\mathbf{A}}{\begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{bmatrix}} \times \underset{\mathbf{B}}{\begin{bmatrix} 7 & 3 & 2 & 1 \\ 8 & 0 & 0 & 2 \\ 1 & 3 & 4 & 0 \end{bmatrix}}$$

Scalar multiplication

$$\underset{\mathbf{B}}{\begin{bmatrix} 15 & 6 & 3 \\ 9 & 6 & 12 \end{bmatrix}} = 3 \times \underset{\mathbf{A}}{\begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{bmatrix}}$$

The determinant of a square matrix A of size m × m denoted as det (A) is a scalar calculated recursively as shown below:

1. If $m = 1$, $\det(\mathbf{A}) = a_{11}$
2. If $m > 1$, $\det(\mathbf{A}) = \sum_{i=1\ldots m} (-1)^{i+j} \times a_{ij} \times \det(\mathbf{A}_{ij})$

Where $\mathbf{A}_{ij}$ is a matrix obtained from $\mathbf{A}$ by deleting the ith row and jth column.

Calculating the determinant of a 2 × 2 matrix based on the determinant of a 1 × 1 matrix

$$\det\begin{bmatrix} 5 & 2 \\ 3 & 4 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det[\,4\,] + (-1)^{1+2} \times 2 \times \det[\,3\,] \longrightarrow 5 \times 4 - 2 \times 3 = 14$$

or

$$\det\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}$$

Calculating the determinant of a 3 × 3 matrix

$$\det\begin{bmatrix} 5 & 2 & 1 \\ 3 & 0 & -4 \\ 2 & 1 & 6 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det\begin{bmatrix} 0 & -4 \\ 1 & 6 \end{bmatrix} + (-1)^{1+2} \times 2 \times \det\begin{bmatrix} 3 & -4 \\ 2 & 6 \end{bmatrix} + (-1)^{1+3} \times 1 \times \det\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

$$= (+1) \times 5 \times (+4) \quad + \quad (-1) \times 2 \times (24) \quad + \quad (+1) \times 1 \times (3) = -25$$

Cryptography uses residue matrices: matrices where all elements are in Z_n. A residue matrix has a multiplicative inverse if gcd (det(A), n) = 1.

$$\mathbf{A} = \begin{bmatrix} 3 & 5 & 7 & 2 \\ 1 & 4 & 7 & 2 \\ 6 & 3 & 9 & 17 \\ 13 & 5 & 4 & 16 \end{bmatrix} \qquad \mathbf{A}^{-1} = \begin{bmatrix} 15 & 21 & 0 & 15 \\ 23 & 9 & 0 & 22 \\ 15 & 16 & 18 & 3 \\ 24 & 7 & 15 & 3 \end{bmatrix}$$

$$\det(\mathbf{A}) = 21 \qquad \det(\mathbf{A}^{-1}) = 5$$

GROUPS, RINGS, AND FIELDS

Groups, rings, and fields are the fundamental elements of a branch of mathematics known as abstract algebra, or modern algebra.

GROUPS

A **group** *G*, sometimes denoted by {*G*, •}, is a set of elements with a binary operation denoted by • that associates to each ordered pair (*a*, *b*) of elements in *G* an element (*a* • *b*) in *G*, such that the following axioms are obeyed:

(A1) Closure: If *a* and *b* belong to *G*, then *a* • *b* is also in *G*.
(A2) Associative: *a*•(*b*•*c*) = (*a*•*b*)•*c* for all *a*, *b*, *c* in *G*.
(A3) Identity element: There is an element e in *G* such that *a* • e = e • *a* = *a* for all *a* in *G*.
(A4) Inverse element: For each *a* in *G*, there is an element a^{-1} in *G* such that $a \bullet a^{-1} = a^{-1} \bullet a$ = e.
If a group has a finite number of elements, it is referred to as a **finite group**, and the **order** of the group is equal to the number of elements in the group. Otherwise, the group is an **infinite group**. A group is said to be **abelian** if it satisfies the following additional condition:
(A5) Commutative: *a* • *b* = *b* • *a* for all *a*, *b* in *G*.
A group *G* is **cyclic** if every element of *G* is a power a^k (*k* is an integer) of a fixed element *a* ε *G*. The element *a* is said to **generate** the group *G* or to be a **generator** of G. A cyclic group is always abelian and may be finite or infinite.

RINGS

A **ring** *R*, sometimes denoted by {*R*, +, *}, is a set of elements with two binary operations, called *addition* and *multiplication*,6 such that for all *a*, *b*, *c* in *R* the following axioms are obeyed.
(A1–A5) *R* is an abelian group with respect to addition; that is, *R* satisfies axioms A1 through A5.
(M1) Closure under multiplication: If *a* and *b* belong to *R*, then *ab* is also in *R*.
(M2) Associativity of multiplication: *a*(*bc*) = (*ab*)*c* for all *a*, *b*, *c* in *R*.

(M3) Distributive laws: $a(b + c) = ab + ac$ for all a, b, c in R.

$(a + b)c = ac + bc$ for all a, b, c in R.

A ring is said to be **commutative** if it satisfies the following additional condition:

(M4) Commutativity of multiplication: $ab = ba$ for all a, b in R.

An **integral domain**, which is a commutative ring that obeys the following axioms.

(M5) Multiplicative identity: There is an element 1 in R such that $a1 = 1a = a$ for all a in R.

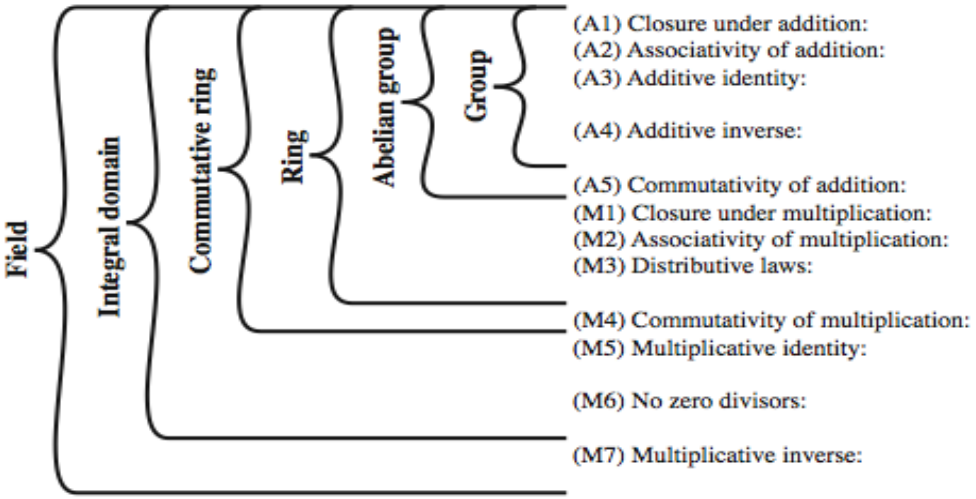
(M6) No zero divisors: If a, b in R and $ab = 0$, then either $a = 0$ or $b = 0$.

FIELDS

A **field** F, sometimes denoted by $\{F, +, *\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in F the following axioms are obeyed.

(A1–M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(M7) Multiplicative inverse: For each a in F, except 0, there is an element a-1 in F such that $aa^{-1} = (a^{-1})a = 1$



FINITE (GALOIS) FIELDS

- Finite fields play a key role in cryptography.
- Finite field is a field that contains a finite number of elements
- It Can show number of elements in a finite field **must** be a power of a prime p^n
- known as Galois fields, denoted GF(p^n)
- In particular often use the fields:

n=1 then we say as GF(p), or p=2 then we say as GF(2^n).

- GF(p) is the set of integers {0,1, … , p-1} with addition & multiplication modulo p.
- This forms a "well-behaved" finite field.

DATA ENCRYPTION STANDARD

Introduction:

The most widely used private key block cipher, is the Data Encryption Standard (DES).

DES encrypts data in 64-bit blocks using a 56-bit key.

IBM developed Lucifer cipher

- ★ by team led by Feistel in late 60's
- ★ used 64-bit data blocks with 128-bit key

- then redeveloped as a commercial cipher with input from National Security Agency (NSA) and others.
- IBM submitted their revised Lucifer which was eventually accepted as the DES.

Over View of DES:

The overall scheme for DES encryption is illustrated:

Plain text must be 64 bits in length and key length is 56 bits in length.

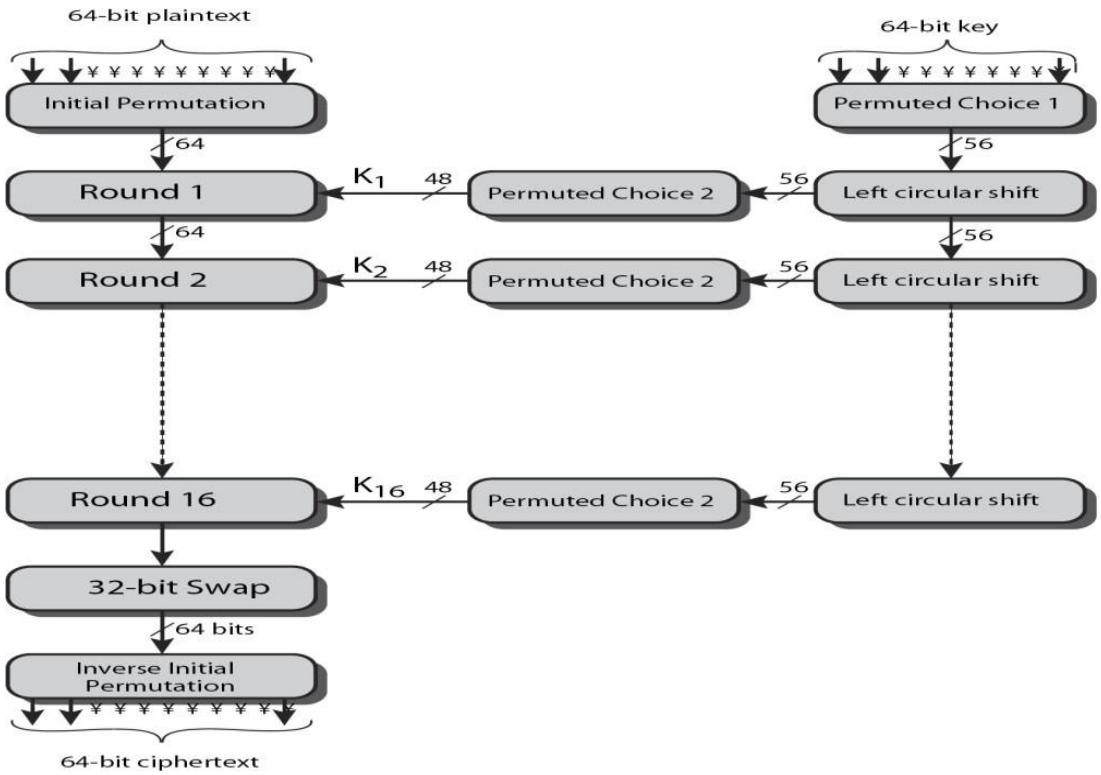
In the figure given below, the left side shows the basic process for enciphering a 64-bit data block which consists of:

- an initial permutation (IP) which shuffles the 64-bit input block
- 16 rounds of a complex key dependent round function involving substitutions & permutations
- a final permutation, being the inverse of IP

The right side shows the handling of the 56-bit key and consists of:

- an initial permutation of the key (PC1) which selects 56-bits out of the 64-bits input, in two 28-bit halves
- 16 stages to generate the 48-bit subkeys using a left circular shift and a permutation of the two 28-bit halves

Fig 1.1 Overview of DES



The main phases in the left hand side of the above figure i.e. processing of the plain text are

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

Table 1: Actual 64 bit order

Initial Permutation (IP): The plaintext block undergoes an initial permutation. 64 bits of the block are permuted. For example the actual input message order can be as follows according to their bit positions:

An example initial Permutation obtained by just writing the even numbered columns first in the row wise and odd numbered columns later in the row wise.

Table 2: Initial Permutation

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

A Single Round Transformation:

- On the right hand side part of the figure, the usage of the 56 bit key is shown. Initially the key is passed through a permutation function.
- Now for each of the 16 iterations, a new subkey (Ki) is produced by combination of a left circular shift and a permutation function which is same for each iteration.
- A different subkey is produced because of repeated shifting of the key bits.
- The left and right halves of each 64 bit intermediate value are treated as separated 32-bit quantities labeled L (left) and R (Right).
- The overall processing at each iteration is given by following steps, which form one round in an S-P network.

$$L_i = R_{i-1}.$$

$$R_i = L_{i-1} \quad F(R_{i-1}, K_i)$$

Where Function F can be described as $P(S(E(R_{i-1}) \oplus K_i))$

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

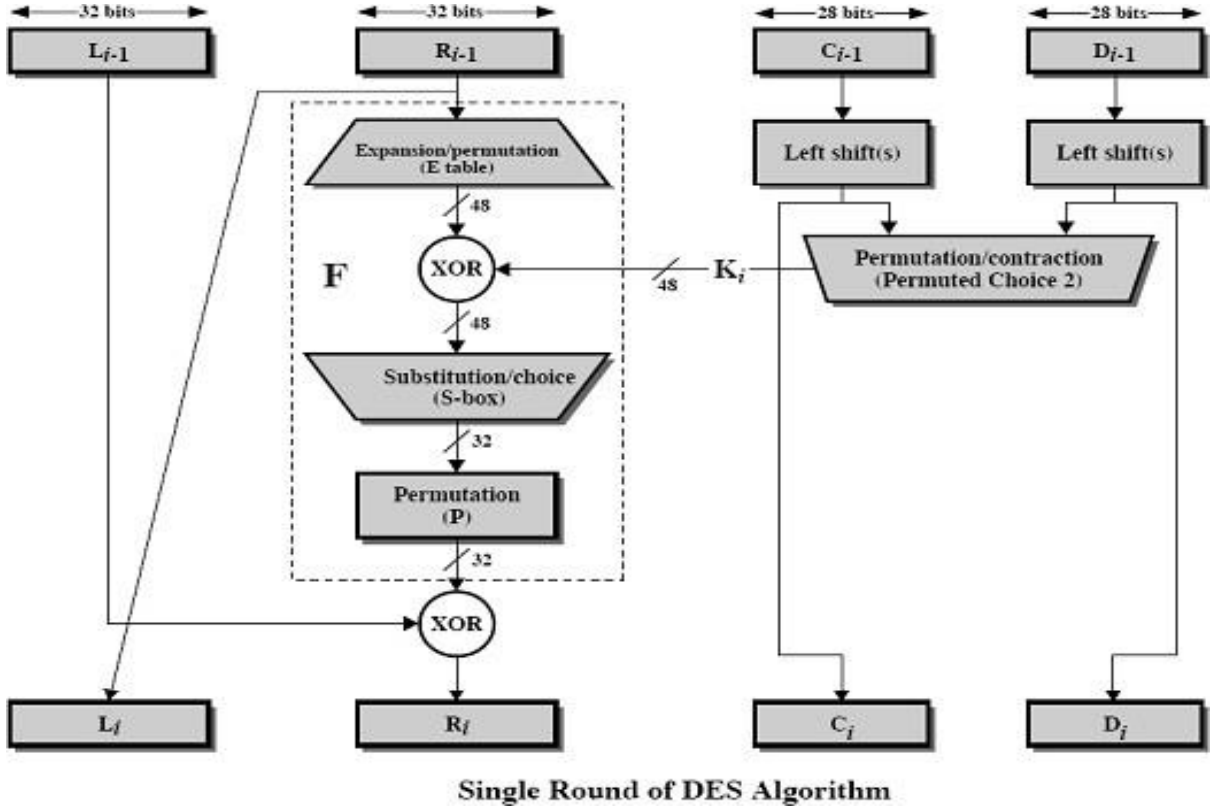


Fig 1.2 Single Round Function

A Complex Transformation (F):

This again contains two sub transformations i) Expansion transformation (E)

ii) Substitution transformation (S)

i) Expansion transformation

- 64 bit permuted block undergoes 16 rounds of complex transformation.
- Subkeys are used in each of the 16 iterations.
- The round Key Ki is 48 bits.
- The R input is 32 bits. The R input is first expanded to 48 bit by using a table that defines a permutation plus an expansion that involves duplication of 16 of R bits.
- The Expansion is as follows: The middle elements belong to the 32 bit R input and Left and right 8 elements are the duplications of the R input, to make as 48 bit.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Table 3: Expansion Permutation

ii) Substitution transformation

The **Role of the S-Box** in the transformation (F) is as follows,

The substitution consists of a set of 8 S-Boxes, each accepts 6 input and produce 4 output bits.

These transformations are defined as follows:

- The first and last bits of 6bits used to specify the row, and the rest of the 4 bits are used to specify the column of the specific box. The data in the specified position is used to replace.
- The first and last bits of the input to box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si.
- The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output.

Table 4: Substitution Table (S-Box)

	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10	Col 11	Col 12	Col 13	Col 14	Col 15
Row0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
Row1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
Row2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
Row3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

For example:

In S_1, for input **011100**, the row is 01 (row 1) and the column is 1100 (column 12).

The value in row 1, column 12 is 9, so the output is 1001.

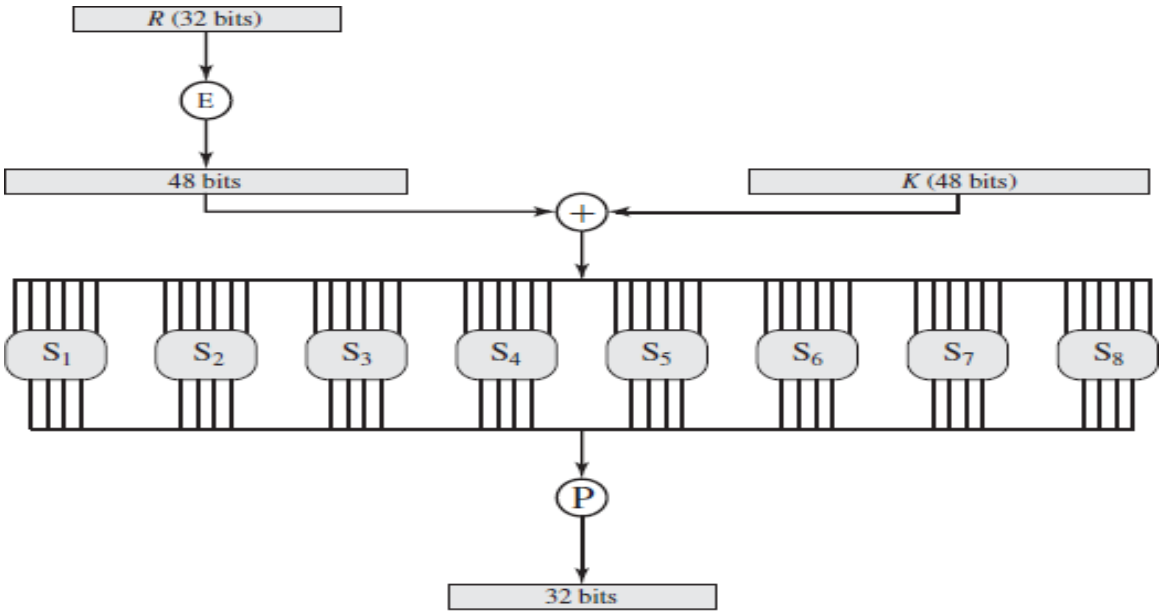


Figure 3.7 Calculation of F(R, K)

The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

Inverse Initial Permutation (IP^{-1}): The 64 bit output undergoes a permutation that is inverse of the initial permutation.

Table 5: Final Permutation

(b) Inverse Initial Permutation (IP^{-1})

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

Sub Key Generation Algorithm: A 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; **every eighth bit is ignored, as indicated by lack of shading** in Table 6. The key is first subjected to a permutation governed by a table labelled Permuted Choice One, indicated in Table 7.

The resulting 56-bit key is then treated as two 28-bit quantities, labelled C_0 and D_0. **At each round, Ci-1 and Di-1 are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by Table 9.** These shifted values serve as input to the next round. They also serve as input to the part labelled Permuted Choice Two, represented in Table 8, which produces a 48-bit output that serves as input to the function .

Table 6: Input Key

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

Table 7: Permuted Choice-1

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

Table 8: Permuted Choice-2

14	17	11	24	1	5	3	28
15	6	21	10	23	18	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

Table 9: Schedule of Left Shifts

Round Numbers	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

.

DES Decryption:

The DES decryption uses the same algorithm as encryption, except the application of sub key in the reverse order.

The Avalanche Affect of DES:

A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext. In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the avalanche effect.

Strength of DES – Key Size

1. The Key

- **The level of security provided by DES in two areas: key size and the nature of the algorithm.**
- **With a key length of 56 bits, there are 2^56 possible keys, which is approximately 7.2*10^16 keys. Thus a brute-force attack appeared impractical**. However DES was finally and definitively proved insecure in July 1998. W
- It is important to note that there is more to a key-search attack than simply running through all possible keys. Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext. Clearly must now consider alternatives to DES, the most important of which are AES and triple DES.

2. **The Nature of the Encryption algorithm**

The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration.

These techniques utilize some deep structure of the cipher by gathering information about encryptions so that eventually you can recover some/all of the sub-key bits, and then exhaustively search for the rest if necessary.

Generally these are statistical attacks which depend on the amount of information gathered for their likelihood of success. Attacks of this form include ***differential cryptanalysis. Linear cryptanalysis, and related key attacks***.

3. The Timing Attack

A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs. The AES analysis process has highlighted this attack approach, and showed that it is a concern particularly with smartcard implementations, though DES appears to be fairly resistant to a successful timing attack.

TRIPLE DES

Double DES:

The simplest form of multiple encryption has two encryption stages and two keys, Fig 2.1 Given a plaintext P and two encryption keys K1 andK2, ciphertext is generated as

$$C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$$

Decryption requires that the keys be applied in reverse order:

$$P = \mathrm{D}(K_1, \mathrm{D}(K_2, C))$$

For DES, this scheme apparently involves a key length of 56 X 2=112 bits, resulting in a dramatic increase in cryptographic strength. But we need to examine the algorithm more closely.

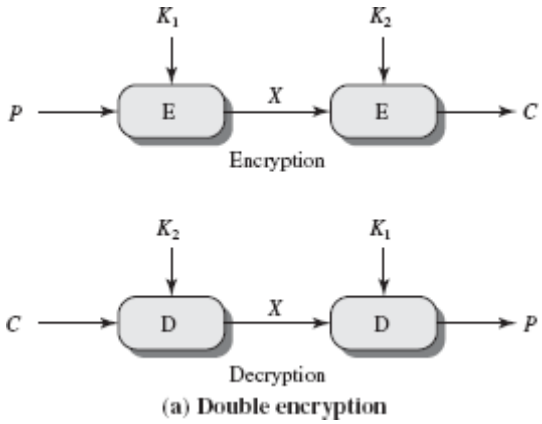


Fig 2.1 Double DES

The first answer to problems of DES is an algorithm called Double DES which includes double encryption with two keys. It increases the key size to 112 bits, which seems to be secure. But, there are some problems associated with this approach.

- ***Issue of reduction to single stage***:

Suppose it were true for DES, for all 56-bit key values, that given any two keys K1 and K2, it would be possible to find a key K3 such that

$$\mathrm{E}(K_2, \mathrm{E}(K_1, P)) = \mathrm{E}(K_3, P)$$

If this were the case, then double encryption, and indeed any number of stages of multiple encryption with DES, would be useless because the result would be equivalent to a single encryption with a single 56-bit key.

Meet-in-the-middle" attack:

Given a known pair (P, C), the attack proceeds as follows.

First, encrypt P for all 256 possible values of K1. Store these results in a table and then sort the table by the values of X. Next, decrypt C, using all 2^{56} possible values of K2.

As each decryption is produced, check the result against the table for a match. If a match occurs, then test the two resulting keys against a new known plaintext–cipher text pair.

If the two keys produce the correct cipher text, accept them as the correct keys. Test the two keys for the second pair of plaintext-cipher text and if they match, correct keys are found.

Triple DES

Triple DES was the answer to many of the shortcomings of DES. Since it is based on the DES algorithm, it is very easy to modify existing software to use Triple DES. 3 DES was developed in 1999 by IBM – by a team led by Walter Tuchman.

3 DES prevents a meet-in-the-middle attack. 3 DES has a 168-bit key and enciphers blocks of 64 bits. It also has the advantage of proven reliability and a longer key length that

eliminates many of the shortcut attacks that can be used to reduce the amount of time it takes to break DES.

3DES uses three keys and three executions of the DES algorithm. The function follows an encrypt-decrypt-encrypt (EDE) sequence.

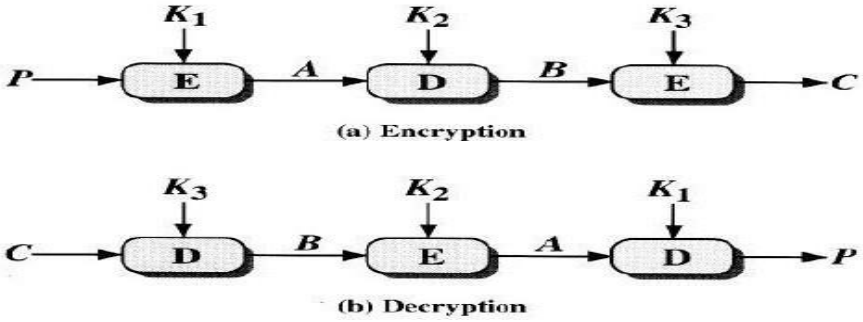


Fig 2.2 Triple DES Encryption and Decryption

C=Ek3[Dk2[Ek1[p]]] Where C= ciphertext, P=plaintext and EK[X] = encryption of X using key K DK[Y] = decryption of Y using key K. Decryption is simply the same operation with the keys reversed.P=Dk1[Ek2[Dk3[c]]]

Triple DES runs three times slower than standard DES, but is much more secure if used properly. With three distinct keys, TDEA has an effective key length of 168 bits making it a formidable algorithm. As the underlying algorithm is DEA, it offers the same resistance to cryptanalysis as is DEA. Triple DES can be done using 2 keys or 3 keys.
There is no cryptographic significance to the use of decryption for the second stage of 3DES encryption. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES:

c = Ek1 [D2l [EK3 [P]]] = EK1 [P]

Strength of Triple DES:

If we assume that the cracker would perform 1 million decryptions per 1micro second, then DES would take 10 hours to break. With 128 bit Key It would take 1018 years to break. With 168 bit key Brute force attack is impossible.

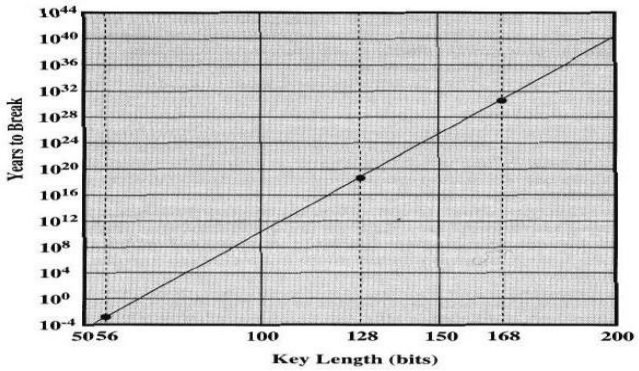


Fig 2.3 Time to break Code

BLOCK CIPHER PRINCIPLES

Any plain text can be transformed into cipher text with following Ciphers

- **Stream Cipher** is the one that encrypts the digital data one bit or one byte at a time to produce the cipher text. Ex:-Autokeyed Vigenere Cipher and Verman Cipher
- **Block Cipher** is the one in which a block of plain text is treated as whole and used to produce a cipher text of equal length. Ex:- DES

Motivations for Feistel Cipher Structure:

- A block cipher operates on a plain text of n-bits to produce the cipher text of n-bits. There are 2^n different possible plain text blocks for encryptions to be reversible (Eg: for decryption to be possible), and each must produce **unique cipher text block. Such transformation is called "reversible" or "nonsingular"**
- **If same cipher block has different plain text block such scheme is called "irreversible Transformation"**

Table 1: Reversible and Irreversible Transformations

Reversible Mapping		Irreversible Mapping	
Plain Text	Cipher Text	Plain Text	Cipher Text
00	11	00	11
10	01	10	01
01	00	01	01

In the above example the same cipher block "01" is producing different plain text blocks; this is called "irreversible transformation".

Block Cipher Principles:

Most symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher.

A block cipher operates on a plaintext block of n bits to produce a cipher text block of n bits. An arbitrary reversible substitution cipher for a large block size is not practical, however, from an implementation and performance point of view.

Feistel points out that what is needed is an approximation to the ideal block cipher system for large n, built up out of components that are easily realizable.

Ideal Block Cipher:

The most general form of the block cipher is Ideal Block Cipher in which a reversible mapping between plain text and cipher text is possible. This allows the maximum number of possible encryption mappings from plain text to cipher text.

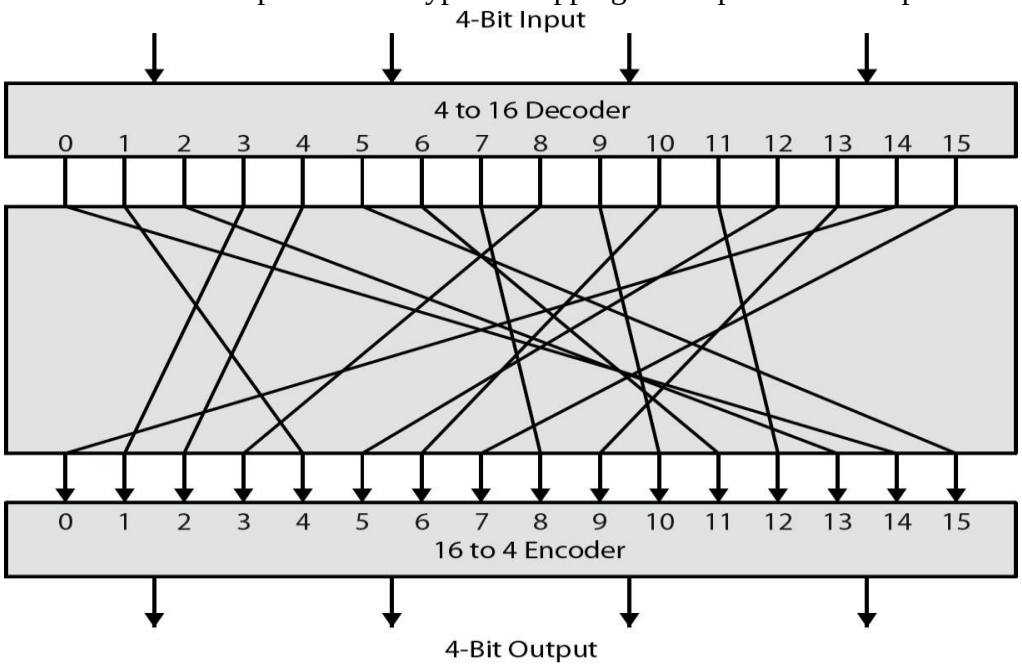


Fig 3.1 General n-bit by n-bit Ideal Block Cipher

Feistel Cipher:

- Horst Feistel, working at IBM Thomas J Watson Research Labs devised a suitable invertible cipher structure in early 70's.
- He proposed that we can actually approximate the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way

that to produce the final result or product that is cryptographically stronger than any of the component ciphers.

- The structure uses the alternate use of substitutions and permutations, which is proposed by the **Claude Shannon** to develop the product cipher that alternates ***confusion*** and ***diffusion***.

Confusion and Diffusion:

- The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system.
- Every block cipher involves a transformation of a block of plaintext into a block of cipher text, where the transformation depends on the key.
- The mechanism of **diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to prevent attempts to deduce the key.
- **Confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.
- So successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern block cipher design.

Feistel Cipher Structure

- Horst Feistel devised the **feistel cipher**
  - based on concept of invertible product cipher
- partitions input block into two halves
  - process through multiple rounds which perform a substitution on left data half
  - based on round function of right half &
  - sub key then have permutation

  swapping halves
- implements Shannon's S-P net concept

The Design Elements of the Feistel Cipher are as follows:

- **Block size** - increasing size improves security, but slows cipher
- **Key size** - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **Number of rounds** –single round offers inadequate security, but multiple rounds increase security, increasing number improves security, but slows cipher
- **Sub key generation** algorithm - greater complexity can make analysis harder, but slows cipher
- **Round function** - greater complexity can make analysis harder, but slows cipher
- **Fast software encryption/decryption –** in many cases encryption is embedded in

hardware to increase speed of cipher, more recent concern for practical use

- **Ease of analysis - for easier validation &** there is a great advantage in designing easier algorithm to find all the vulnerabilities, testing of strength

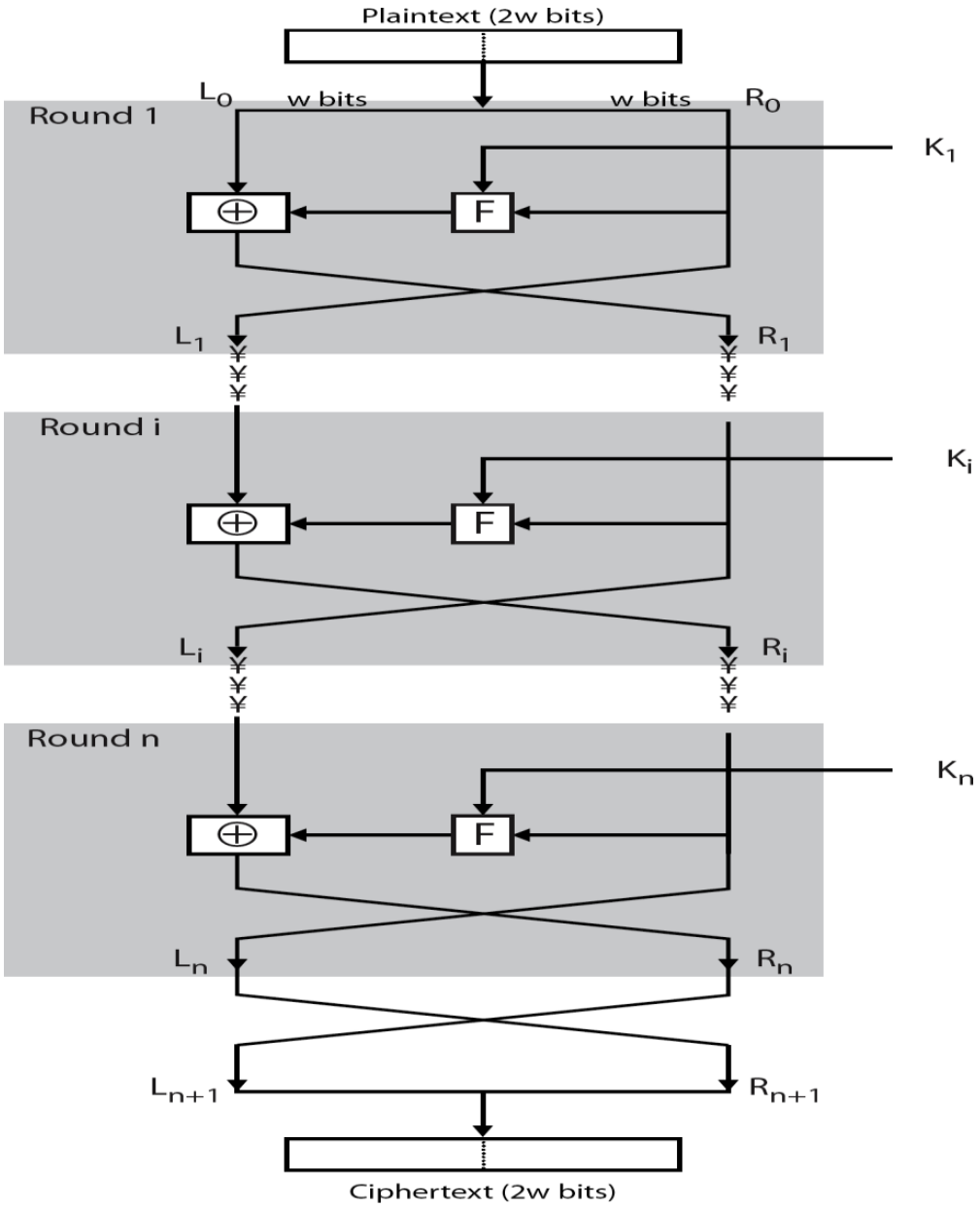


Fig 3.2 Encryption Process

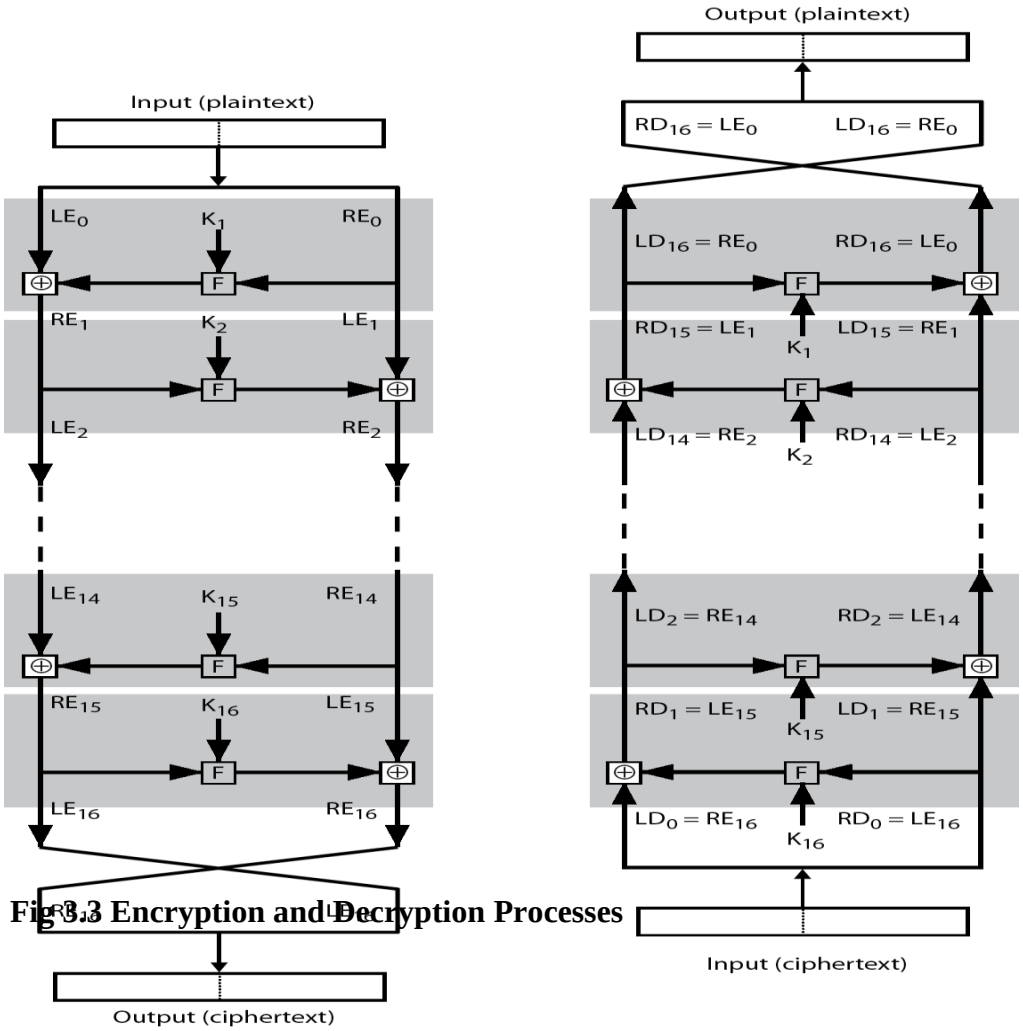


Fig 3.3 Encryption and Decryption Processes

Working procedure of Feistel Cipher:

The plaintext block is divided into two halves, *Lo* and *Ro.*

The two halves of the data pass through *n* rounds of processing and then combine to produce the ciphertext block.

Each round *i* has as inputs *Li-l* and *R;-l'* derived from the previous round, as welI as a subkey Ki, derived from the overall *K.*

In general, the subkeys *Ki* are different from *K* and from each other and are generated from the key by a subkey generation algorithm.

All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR (X OR) of the output of that function and the left half of the data.

The round function has the same general structure for each round but is parameterized by the round subkey *Ki.* Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

The general process of the encryption will be as

follows: $LE_i = RE_{i-1}$
$RE_i = LEI \text{ X } F(RE_{i-1}, K_i)$

The Decryption is as

follows: $LE_{16} = RE_{15}$
$RE_{16} = LE_{15} \text{ X } F(RE_{15}, K_{16})$

Differential Cryptanalysis

One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis. In this section, we discuss the technique and its applicability to DES. The differential cryptanalysis attack is complex.

The rationale behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block.

Consider the original plaintext block **m** to consist of two halves [$\mathbf{m_0, m_1}$. Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the sub key for this round. So, at each round, only one new 32-bit block is created. If we label each new block m_1 ($2 \leq i \leq 17$), then the intermediate message halves are related as follows:

$$\mathbf{m_{i+1} = m_{i-1} \oplus f(m_i, K_i),\ i = 1, 2, ..., 16}$$

This attack is known as Differential Cryptanalysis because the analysis compares differences between two related encryptions, and looks for a known difference in leading to a known difference out with some (pretty small but still significant) probability. If a number of such differences are determined, it is feasible to determine the subkey used in the function f.

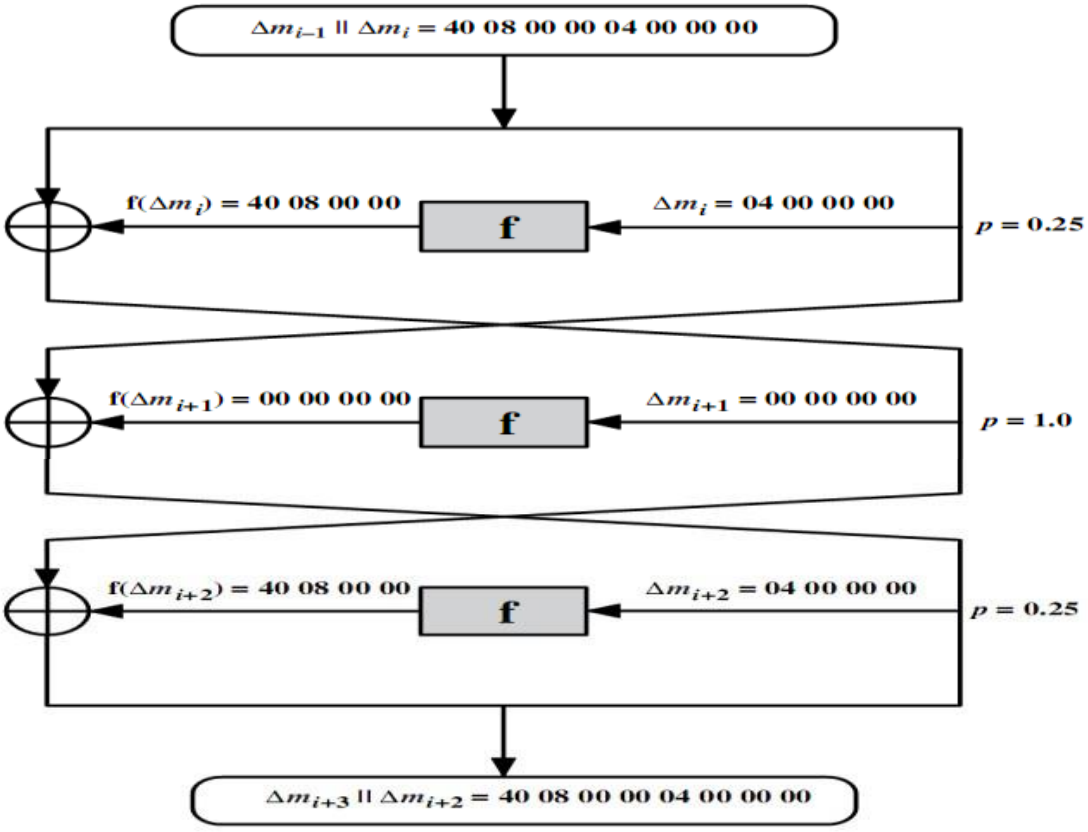


Figure 3.7 Differential Propagation through Three Round of DES (numbers in hexadecimal)

The overall strategy of differential cryptanalysis is based on these considerations for a single round. The procedure is to begin with two plaintext messages m and m' with a given difference and trace through a probable pattern of differences after each round to yield a probable difference for the cipher text. You submit m and m' for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match, then suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, can make some deductions about the key bits. This procedure must be repeated many times to determine all the key bits.

Linear Cryptanalysis

A more recent development is linear cryptanalysis. This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given 243 known plaintexts, as compared to 247 chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES.

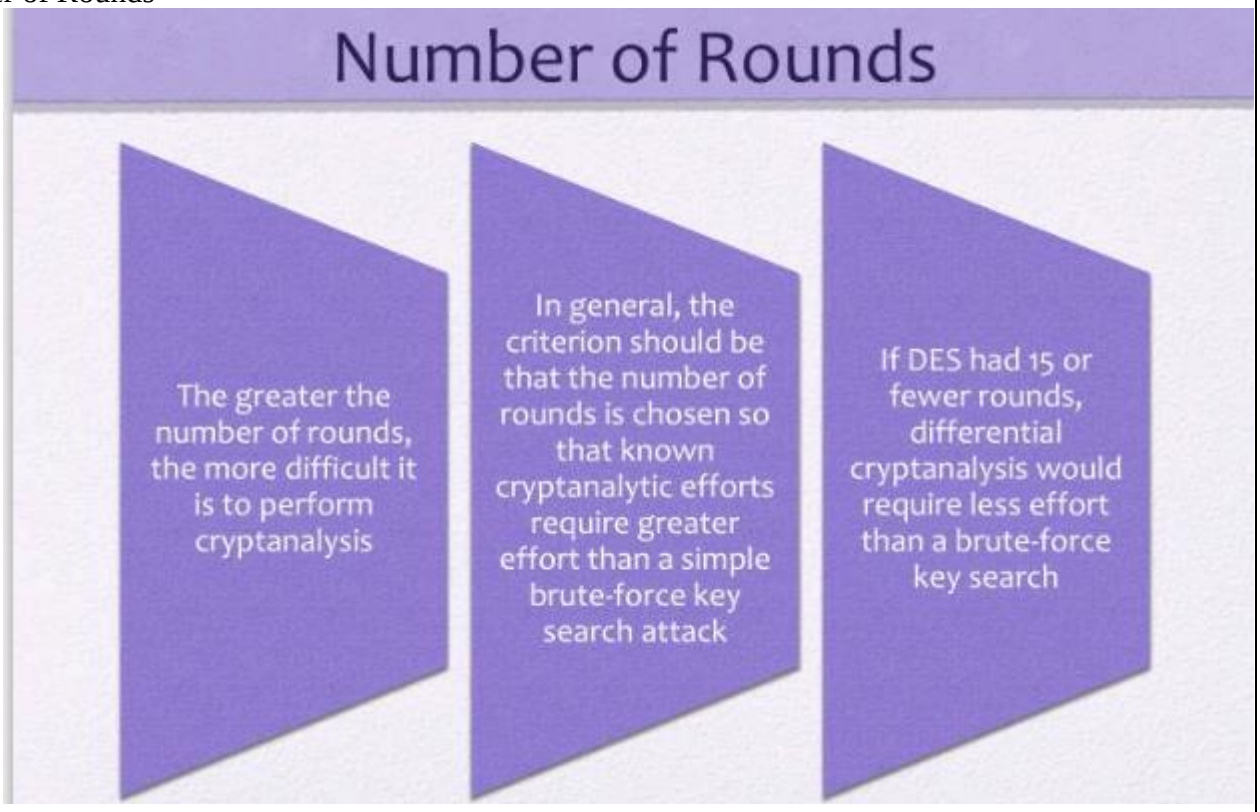
Distinguish between differential and linear cryptanalysis. (May 2012)

- **Linear cryptanalysis** is a general form of cryptanalysis based on finding affine approximations to the action of a cipher. Linear cryptanalysis is one of the most widely used attacks on block ciphers;
- **Differential cryptanalysis** is a general form of cryptanalysis applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in information input can affect the resultant difference at the output.
- **Linear cryptanalysis** focuses on statistical analysis against one round of decrypted cipher text. **Differential** analysis focuses on statistical analysis of two inputs and two outputs of a cryptographic algorithm.

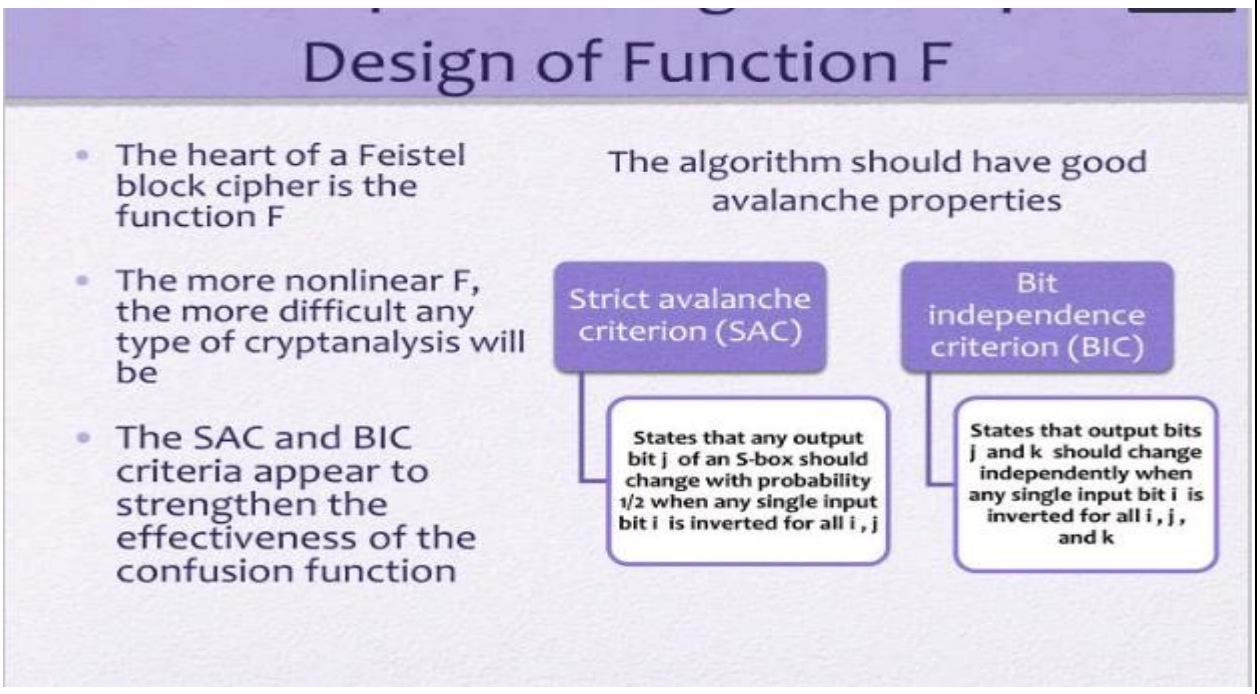
BLOCK CIPHER DESIGN PRINCIPLES

1) Number of Rounds
2) Design of Function F
3) Key Schedule Algorithm

1)Number of Rounds



2)Design of Function F



3)Key Schedule Algorithm

- With any feistel block cipher, the key is used to generate one subkey for each round.
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of deducing individual subkeys and the difficulty of working back to the main key.
- It is suggested that, at a minimum,the key schedule should guarantee key / cipher text strict Avalanche criterion and Bit independence Criterion.

BLOCK CIPHER MODES OF OPERATION

A symmetric block cipher processes one block of data at a time. In the case of DES and 3DES, the block length is 64 bits. For longer amounts of plaintext, it is necessary to break the plaintext into 64-bit blocks (padding the last block if necessary). There are five modes of operations:

- Electronic Codebook Mode
- Cipher Block Chaining Mode
- Cipher Feedback Mode.
- Output Feedback Mode
- Counter Mode

(i) Electronic Codebook Mode:

- The **simplest way** to proceed is what is known as electronic codebook (ECB) mode, in which **plaintext is handled 64 bits at a time and each block of plaintext is encrypted using the same key**.

- The term *codebook* is used because, for a given key, there is a unique ciphertext for every 64-bit block of plaintext.
- Therefore, one can imagine a gigantic codebook in which there is an entry for every possible 64-bit plaintext pattern showing its corresponding ciphertext.

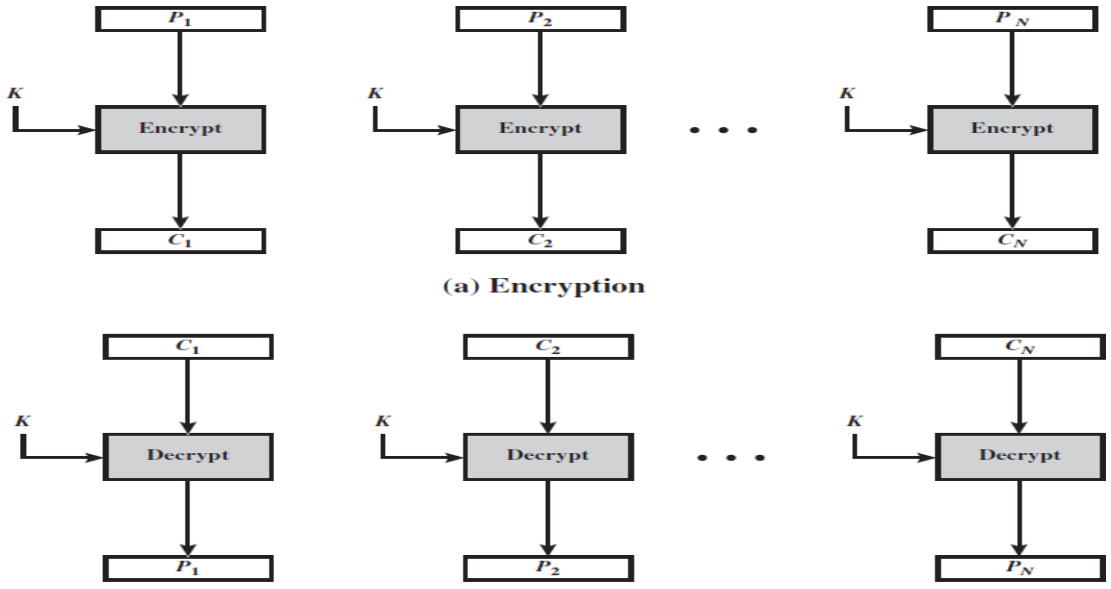


Figure 6.3 Electronic Codebook (ECB) Mode

Advantages and Limitations of ECB:

- Main use is sending a **few blocks of data** .(i.e) If we want to securely send key of DES or AES we can use this mode to send the key securely
- When two messages which have two blocks of plaintexts in common are encrypted with ECB mode the corresponding cipher text blocks will be the same. **Message repetitions may show in ciphertext.**
- Weakness is due to the **encrypted message blocks being independent**

(ii) Cipher Block Chaining Mode (CBC):

- To overcome the problems of repetitions and order independence in ECB, want some way of making the ciphertext dependent on all blocks before it. This is what CBC gives us, by combining the previous ciphertext block with the current message block before encrypting.
- To start the process, use an Initial Value (IV), which is usually well known (often all 0's).
- CBC mode is applicable whenever large amounts of data need to be sent securely, provided that all data is available in advance (eg email, FTP, web etc).

- In effect, we have chained together the processing of the sequence of plaintext blocks, to avoid the **similarities**. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of 64 bits are not exposed.
- The Encryption process is as follows:

  Ci=Ek[Ci-1 XOR Pi], where Ek[x] is the encryption of plain text x, using the key K, and XOR is the Exclusive OR operation.
- The Decryption process is as follows: Pi=Dk [Ci] XOR Ci-1

For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block.

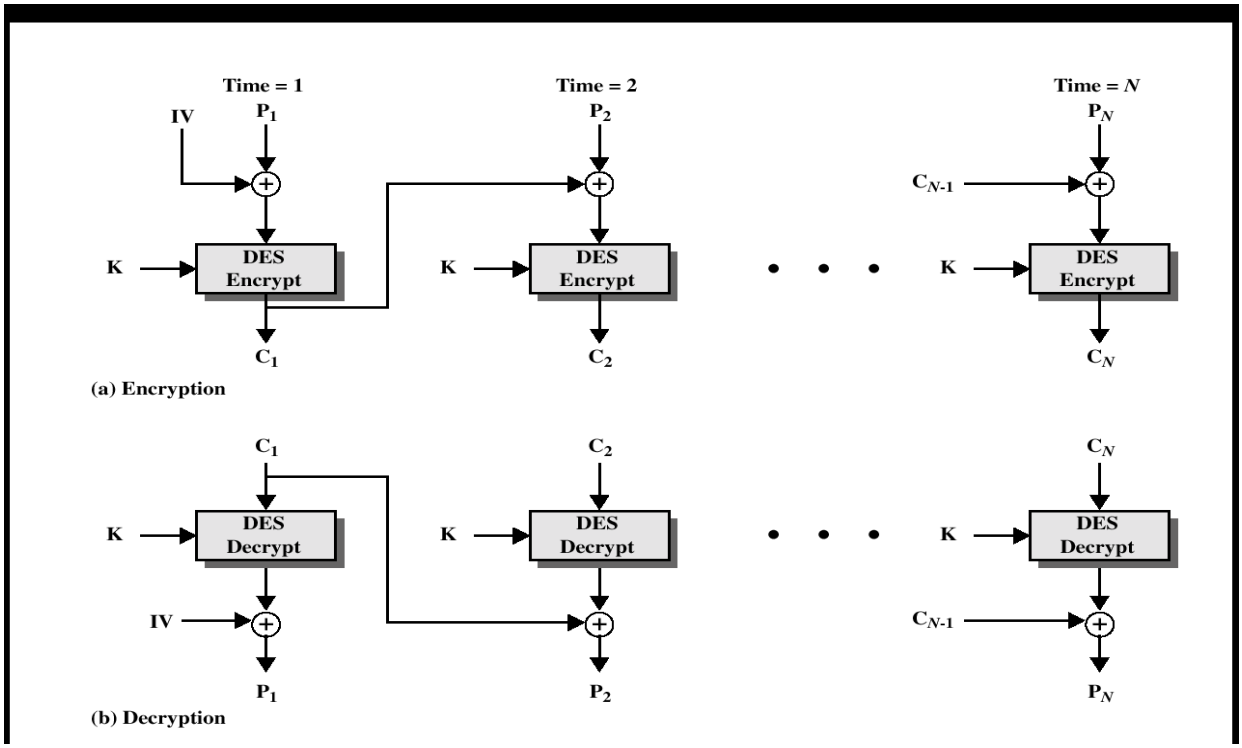


Fig 4.2 Cipher Block Chaining

Mode Advantages and Limitations of CBC:

- A ciphertext block depends on all blocks before it
- Any change to a block affects all following ciphertext blocks
- Need Initialization Vector (IV)
- Padding is done (Last block padded with b bits if it is partial block)

(iii) Cipher Feedback Mode (CFB):

It is possible to convert any block cipher into a stream cipher by using the cipher feedback (CFB) mode.

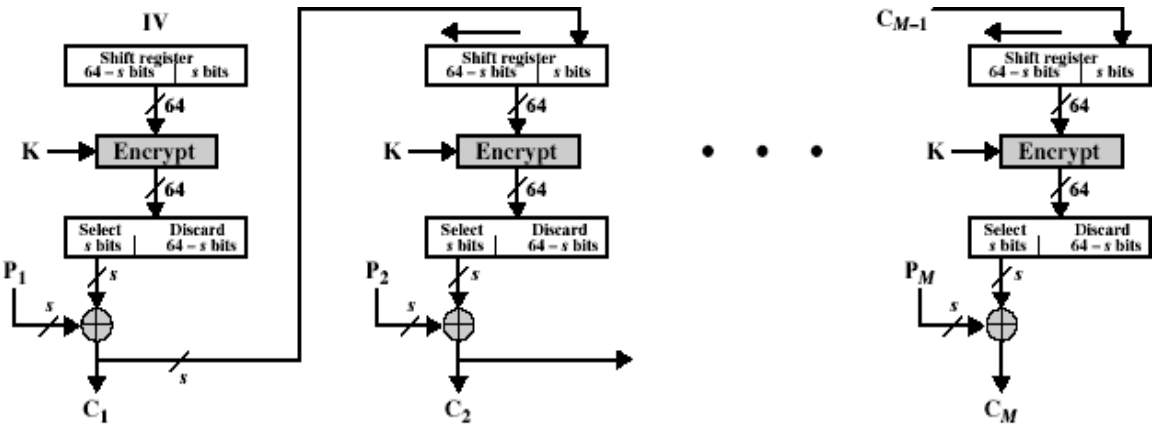
A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character- oriented stream cipher.

First, consider encryption.

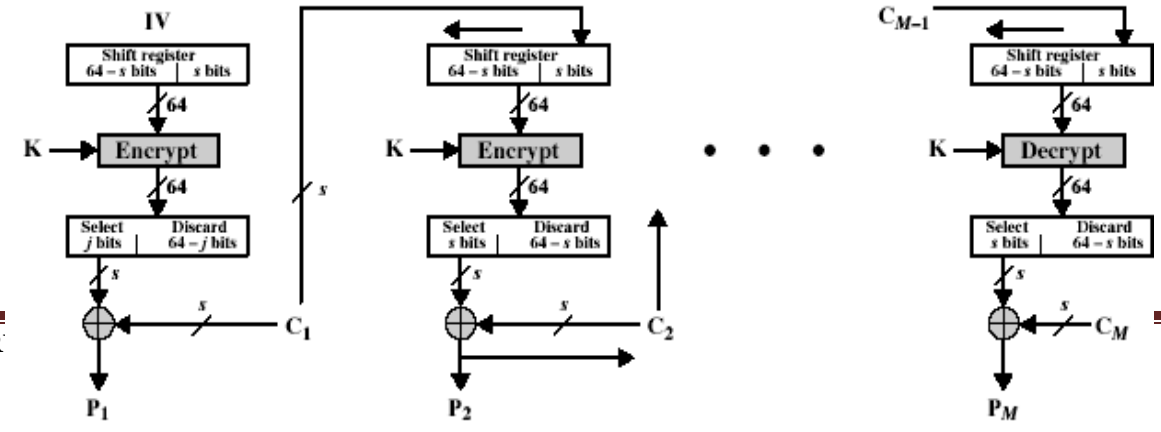
- The input to the encryption function is a 64-bit shift register that is initially set to some initialization vector (IV).
- The leftmost (most significant) *s* bits of the output of the encryption function are XORed with the first unit of plaintext to produce the first unit of ciphertext , which is then transmitted.
- In addition, the contents of the shift register are shifted left by *s* bits and Cipher text is placed in the rightmost (least significant) *s* bits of the shift register. This process continues until all plaintext units have been encrypted.

Mode Advantages and Limitations of CFB:

- Appropriate when data arrives in bits/bytes
- Most common stream mode
- Limitation is need to stall while do block encryption after every n-bits
- Errors propagate for several blocks after the error that if its used over a "noisy" link, then any corrupted bit will destroy values in the current and next blocks



(a) Encryption



(b) Decryption

Fig 4.3 Cipher Feedback

(iv) Output Feedback Mode (OFB):

The **output feedback** (OFB) mode is similar in structure to that of CFB.

1) From the below fig, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB, the ciphertext unit is fed back to the become an input for encrypting the next block.

2) OFB mode operates on full blocks of plaintext and ciphertext, whereas CFB operates on an s-bit subset.

3)Like CBC & CFB, OFB uses IV (Must be Nonce- Number used once)

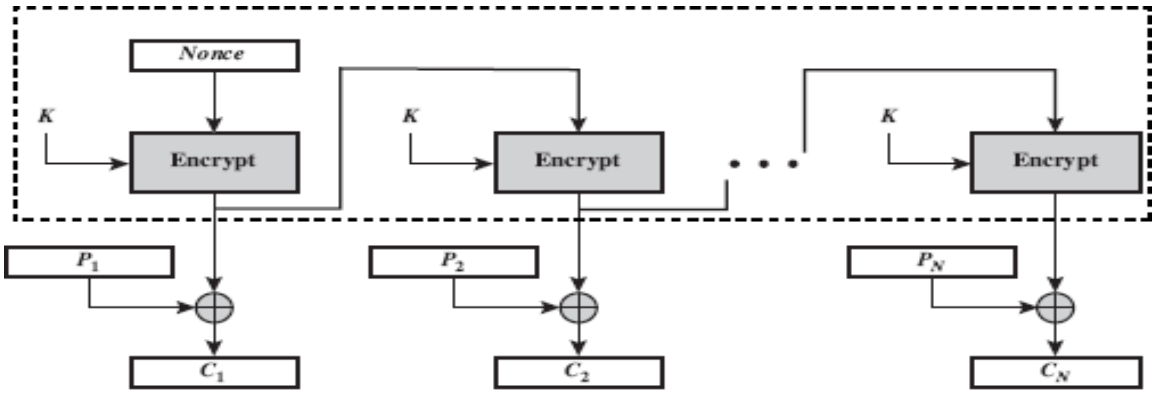
(i.e the IV must be unique to each execution of the encryption algorithm.)

Encryption can be expressed as By rearranging terms,

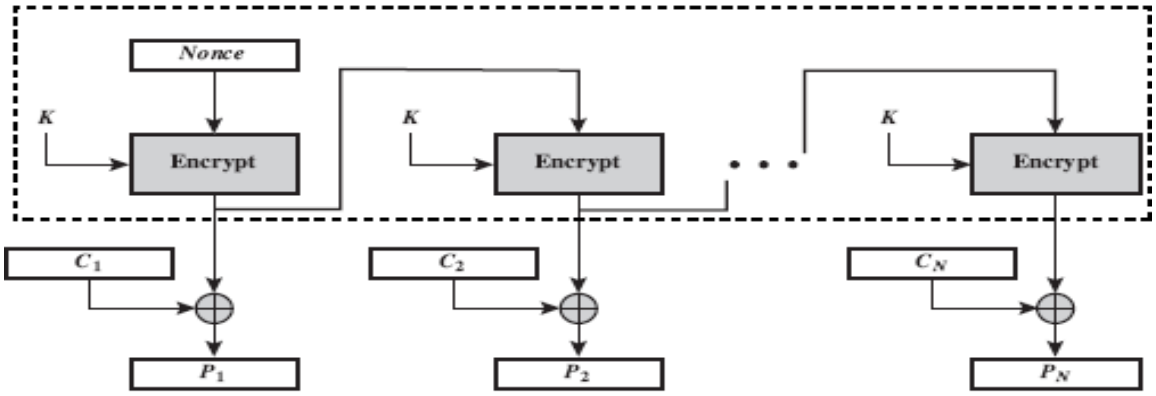
$Cj = Pj\_$ XOR E(K, [Cj-1 XOR Pj-1])

we can demonstrate that decryption works.

$Pj = Cj$ XOR E(K, [Cj-1 XOR Pj-1])



(a) Encryption



(b) Decryption

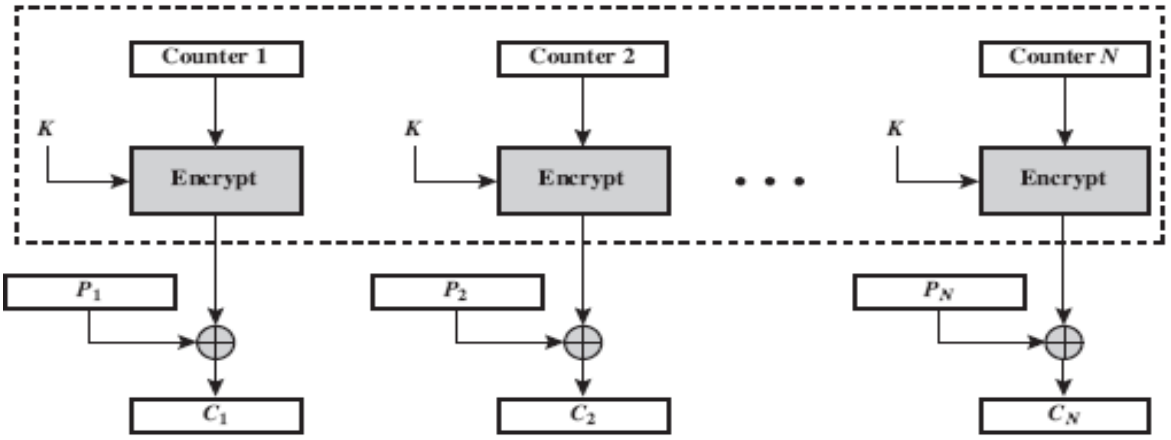
Fig 4.4 Output Feedback Mode

Advantage of OFB Mode: The bit errors in transmission do not propagate (any bit error only affects a single bit)

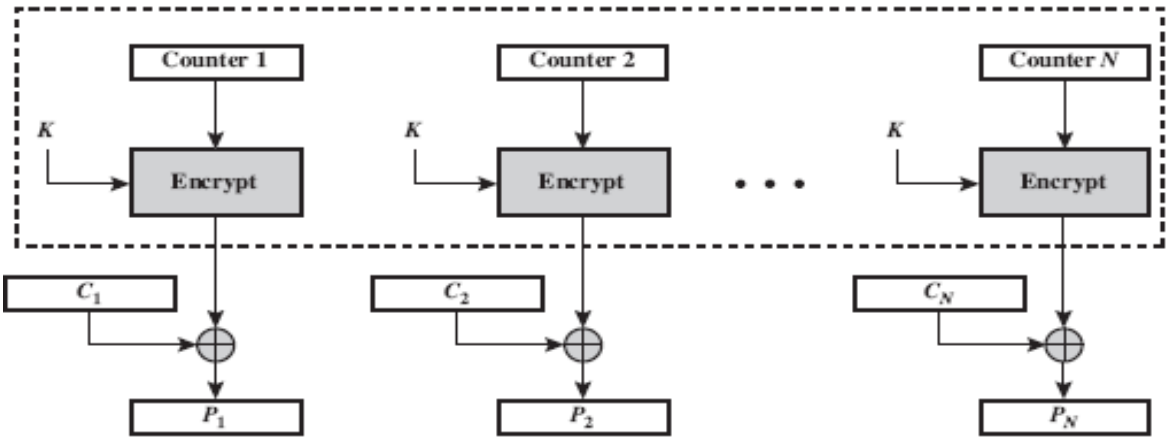
Disadvantage of OFB Mode: The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB.

(v) Counter Mode (CTR):

- Similar to OFB but **encrypts counter value** (hence name) rather than any feedback value
- Must **have a different key & counter value for** every plaintext block (never reused).
- It is being used with applications in ATM (asynchronous transfer mode) network security and IPSec (IP security).
- A counter, equal to the plaintext block size is used.
- Typically the counter is initialized to some value and then incremented by 1 for each subsequent block.



(a) Encryption



(b) Decryption

Fig 4,5 Counter

Mode Advantages and Limitations of CTR:

- Efficiency
  - ★ can do parallel encryptions in h/w or s/w
  - ★ can pre-process in advance of need
  - ★ good for bursty high speed links
- Random access to encrypted data blocks
- Provable security (good as other modes) But must ensure never reuse key/counter values, otherwise could break.

BLOCK CIPHER OPERATION

Table 1 Block Cipher Modes of Operation

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

ADVANCED ENCRYPTION STANDARD

AES Evaluation Criteria:

- Initial criteria:
  - Security – effort for practical cryptanalysis
  - Cost – in terms of computational efficiency

- Algorithm & implementation characteristics

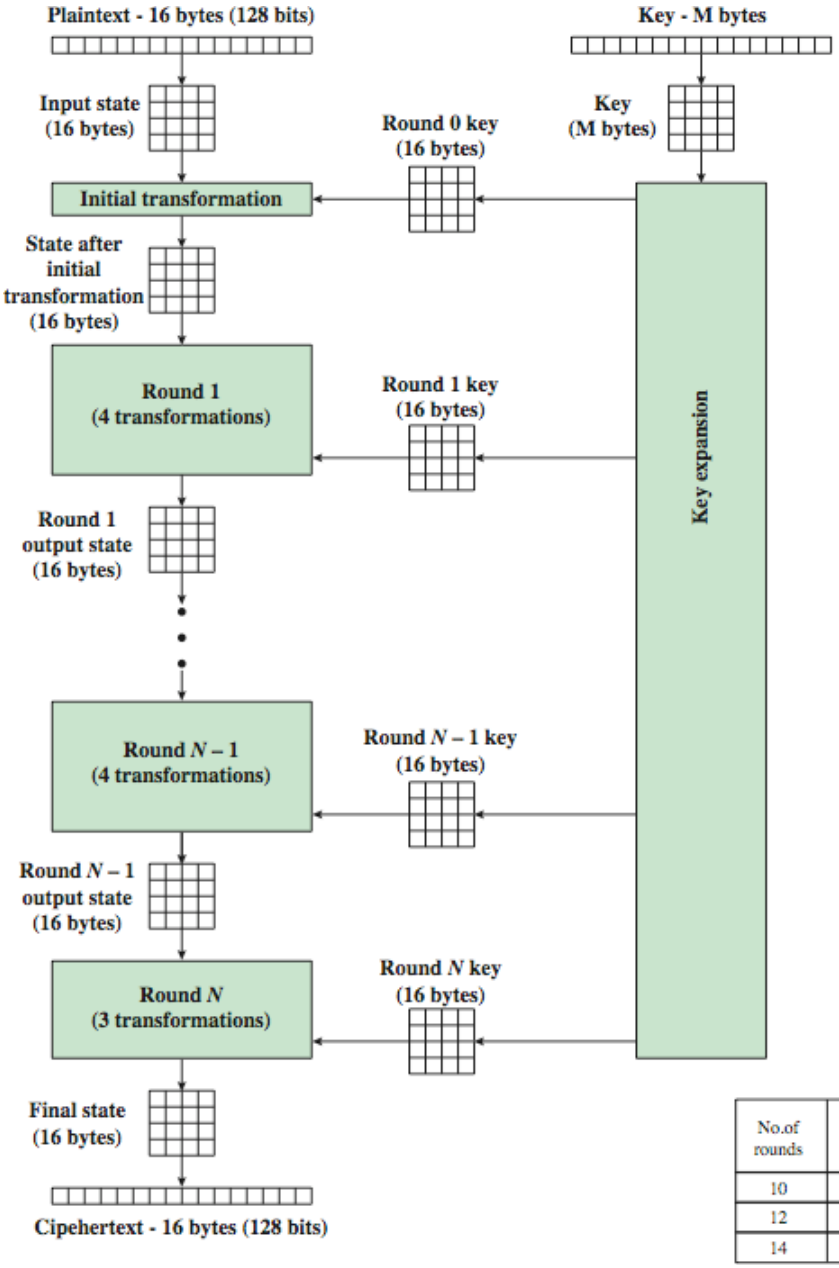
- Final criteria
  - General security
  - Ease of software & hardware implementation
  - Implementation attacks
  - Flexibility (in en/decrypt, keying, other factors)

AES Requirements:

- Private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES

The AES Cipher:

- designed by Rijmen-Daemen in Belgium
- The Advanced Encryption Standard (AES) was published by **NIST** (National Institute of Standards and Technology) in **2001.**
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES ENCRYPTION PROCESS

AESDATASTRUCTURE

Input, state array, and output

The key as a square matrix of bytes expanded into an array of key schedule words. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key.

Key and expanded key

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

Comments about the Overall Structure:

One noteworthy feature of this structure is that it is not a Feistel structure.

The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round. Four different stages are used, one of permutation and three of substitution:

Substitute bytes **:** Uses an S-box to perform a byte-by-byte substitution of the block

ShiftRows : A simple permutation

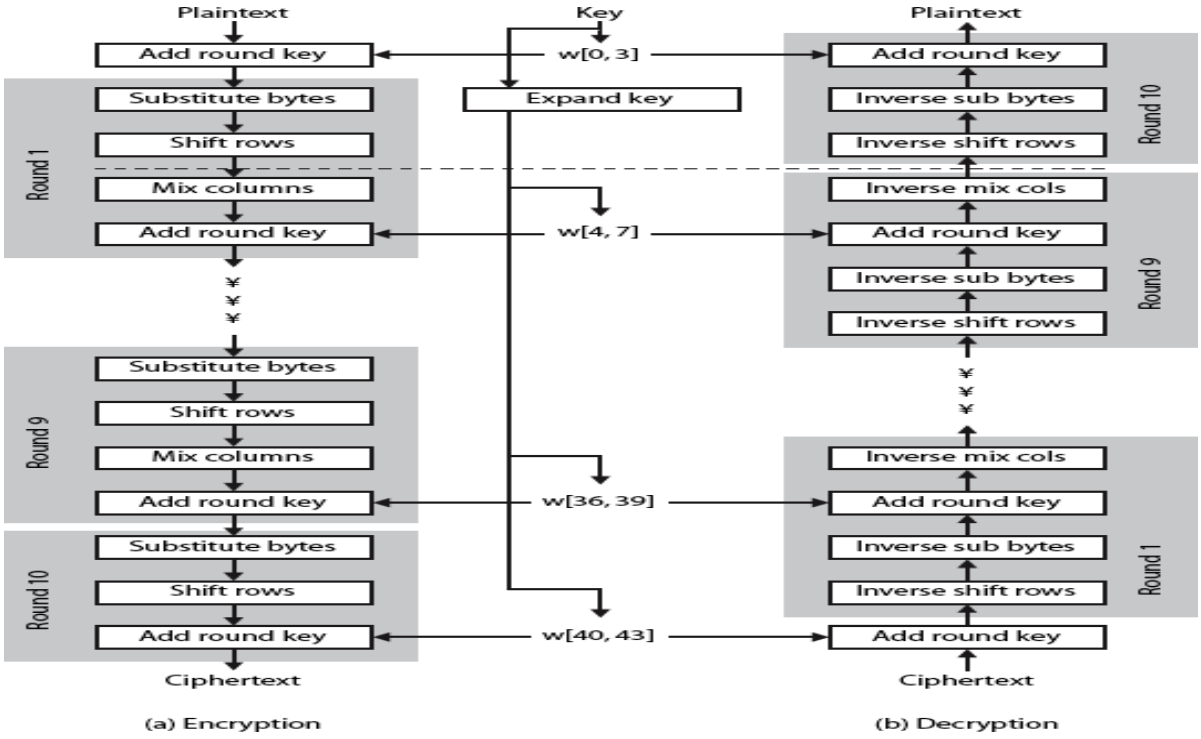
MixColumns **:** A substitution that makes use of arithmetic

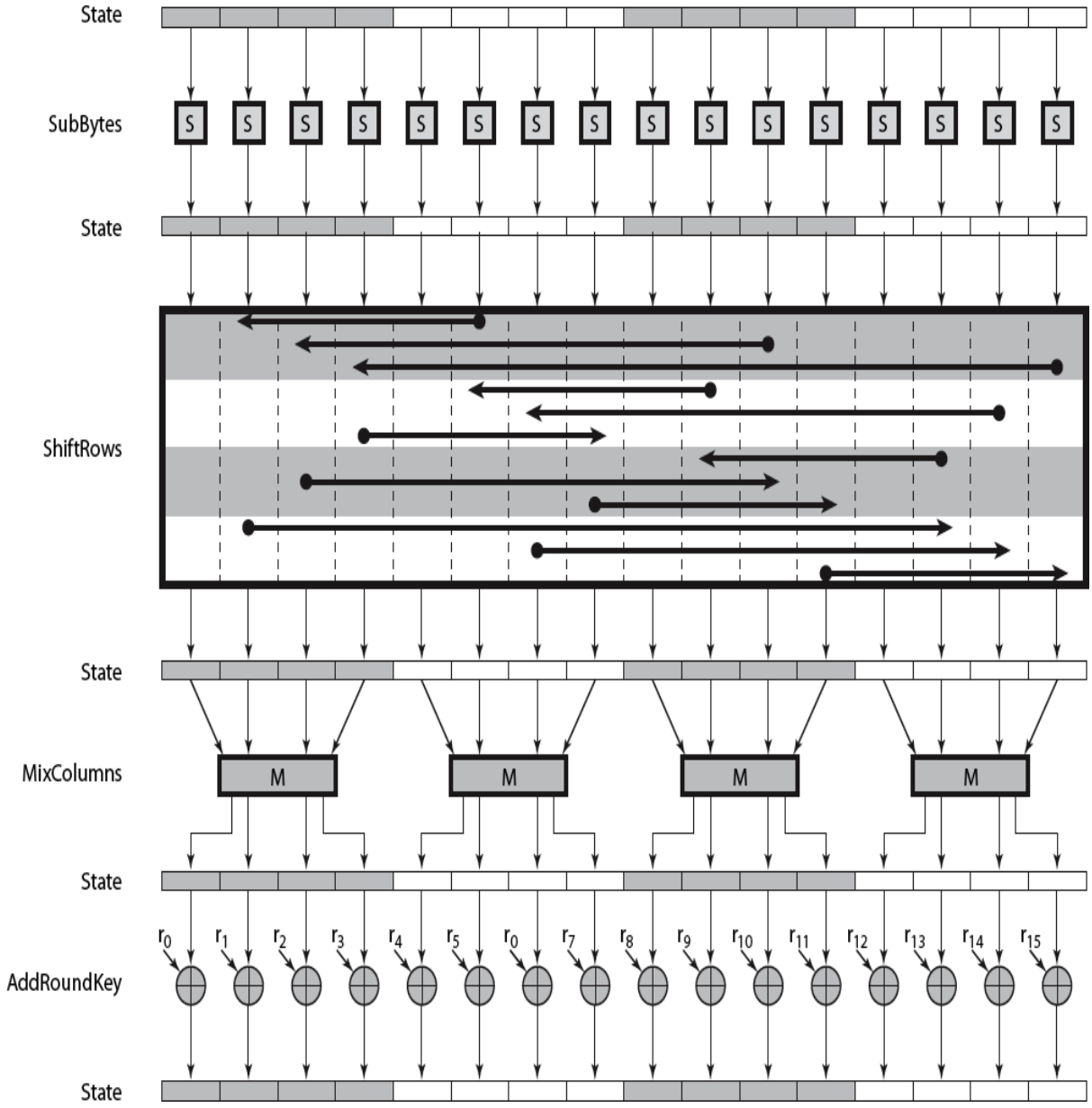
AddRoundKey : A simple bitwise XOR of the current block with a portion of the expanded key

The structure is quite simple. Only the AddRoundKey stage makes use of the key. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. Each stage is easily reversible. As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. Decryption is not identical to the encryption. This is just because of the structure of the AES.

Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. The final round of both encryption and decryption consists of only three stages.

AESSTRUCTURE



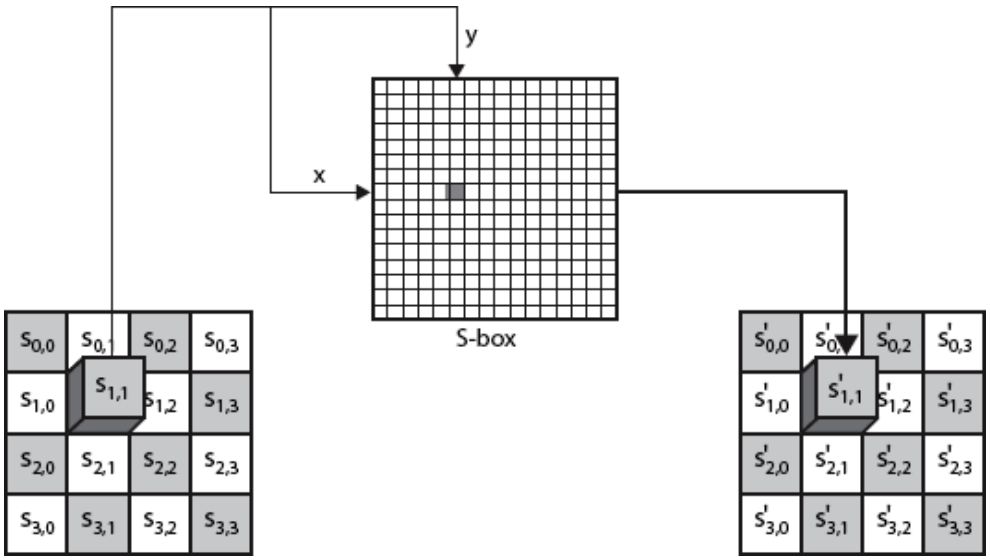


AES ENCRYPTION ROUND

Byte Substitution:

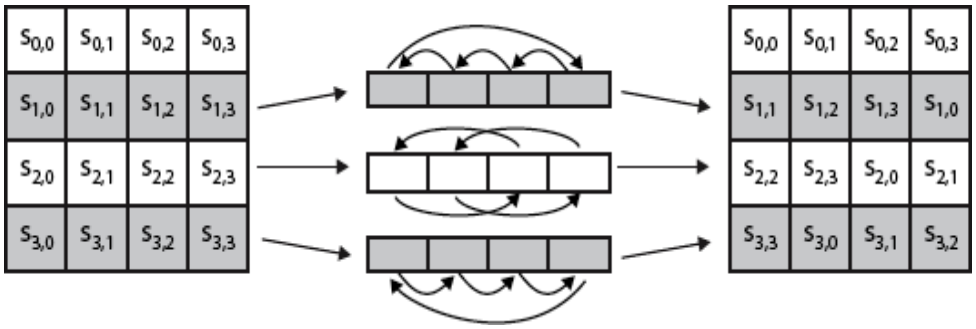
A simple substitution of each byte uses one table of 16x16 bytes containing a permutation of all 256 8-bit values each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)

Eg. byte {95} is replaced by byte in row 9 column 5 which has value {2A}. S-box constructed using defined transformation of values in GF(28) designed to be resistant to all known attacks.



Shift Rows:

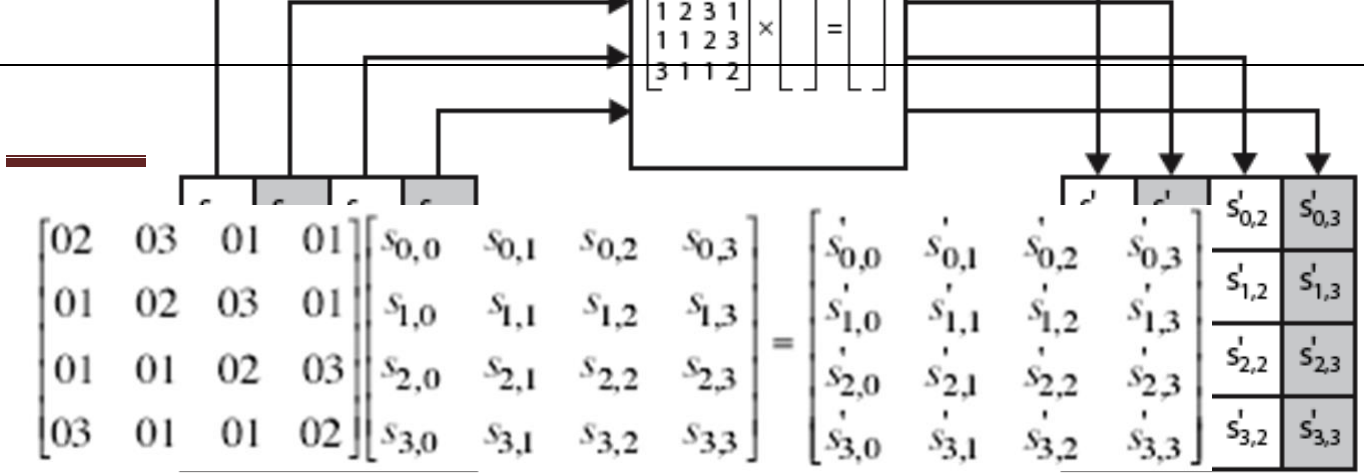
A circular byte shift in each; 1st row is unchanged, 2nd row does 1 byte circular shift to left, 3rd row does 2 byte circular shift to left, 4th row does 3 byte circular shift to left decrypt inverts using shifts to right since state is processed by columns, this step permutes bytes between the columns



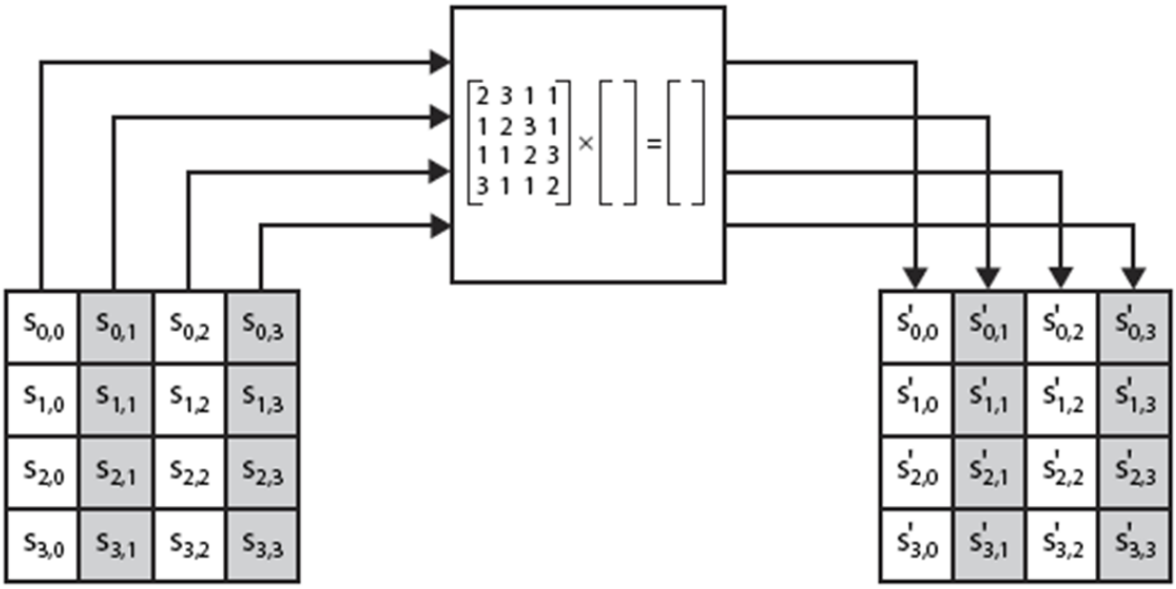
Mix Columns:

Each column is processed separately. Each byte is replaced by a value dependent on all 4 bytes in the column. Effectively a matrix multiplication in GF(28) using prime poly

m(x) =x8+x4+x3+x+1

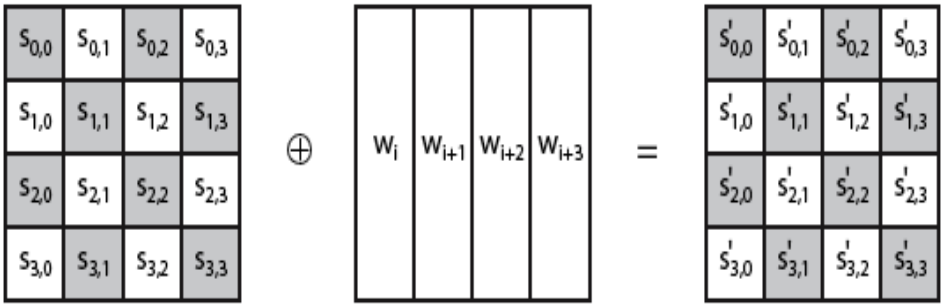


Mix Columns:



AddRoundKey Transformation

The 128 bits of State are bitwise XORed with the 128 bits of the round key. the operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key; it can also be viewed as a byte-level operation. For example:



47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

⊕

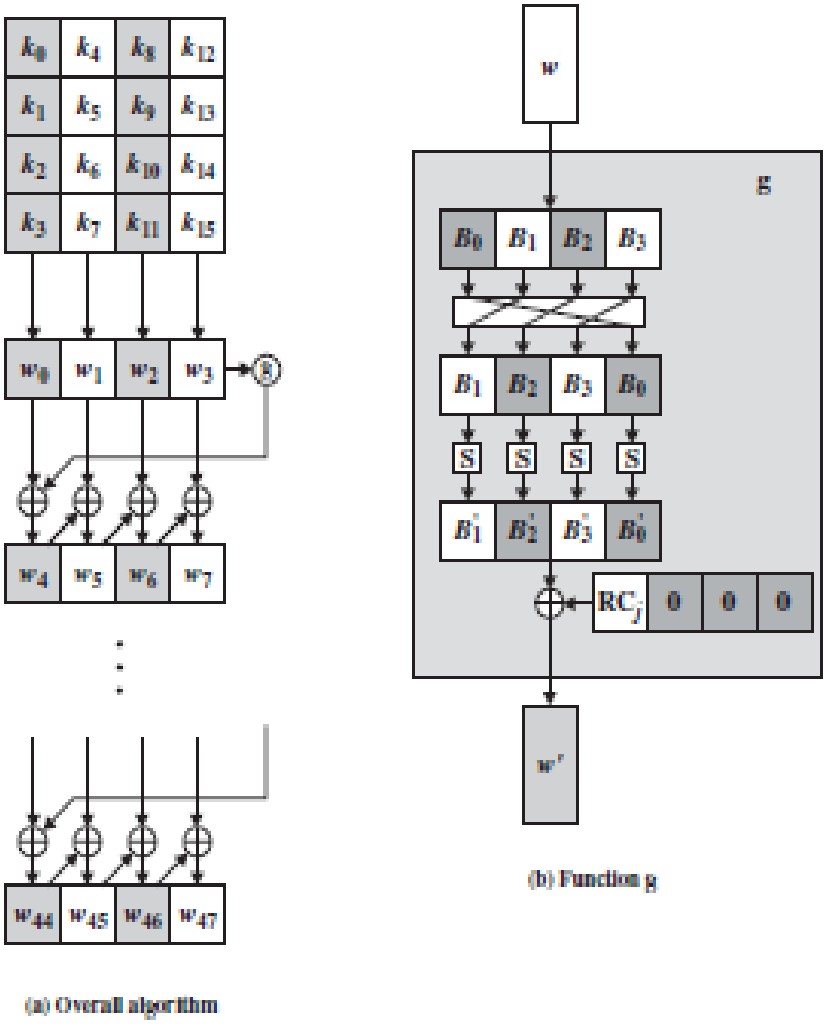
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

AES Key Expansion:

The 128 bit key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. Each added new word depends on the w[i] and w[i-4]. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used. The complex function g consists of the following subfunctions.



(a) Overall algorithm

(b) Function g

Figure 5.9 AES Key Expansion

Where R is the called Rotate Constant.

- RotWord (Rotate Word) performs a one-byte circular left shift on a word.
- SubWord(Substitute Word) performs a byte substitution on each byte of its input word, using the S-box
- The result of steps 1 and 2 is XORed with a round constant, R

RC4

- ✓ RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security.
- ✓ It is a variable key-size stream cipher with byte-oriented operations.
- ✓ The algorithm is based on the use of a random permutation.
- ✓ RC4 is probably the most widely used stream cipher.
- ✓ It is used in the SSL/TLS secure web protocol, & in the WEP & WPA wireless LAN security protocols.
- ✓ RC4 was kept as a trade secret by RSA Security, but in September 1994 was anonymously posted on the Internet on the Cypherpunks anonymous remailers list.
- ✓ the RC4 key is ued to form a random permutation of all 8-bit values, it then uses that permutation to scramble input info processed a byte at a time.

RC4 Key Schedule

- ✓ The RC4 key schedule initialises the state S to the numbers 0..255
- ✓ After doing this 256 times, the result is a well and truly shuffled array.
- ✓ The total number of possible states is 256! - a truly enormous number, much larger even than the 2048-bit (256*8) max key allowed can select.
- ✓ S forms **internal state** of the cipher

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

RC4 Encryption:

To form the stream key for en/decryption (which are identical), RC4 continues to shuffle the permutation array S by continuing to swap each element in turn with some other entry, and using the sum of these two entry values to select another value from the permutation to use as the stream key, which is then XOR'd with the current message byte.

```
i = j = 0
```

for each message byte M_i

i = (i + 1) (mod 256)

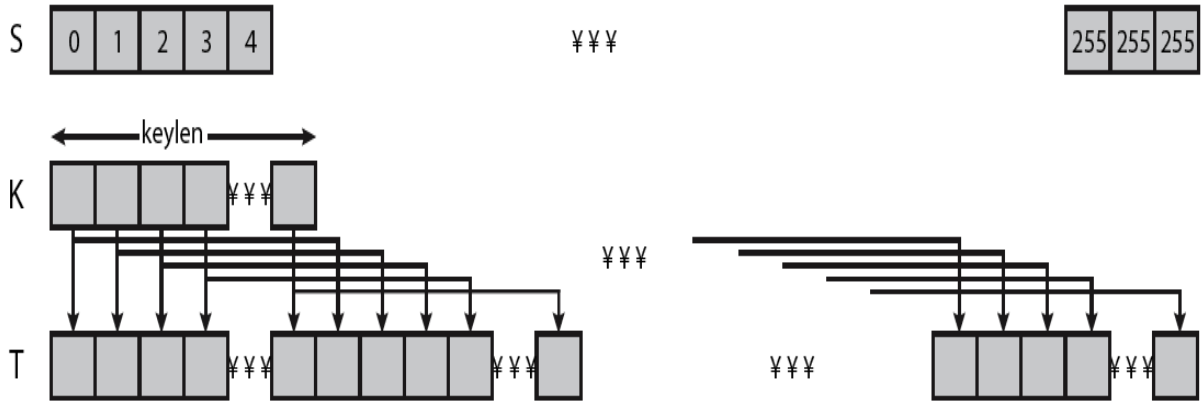
j = (j + S[i]) (mod 256)

swap(S[i], S[j])

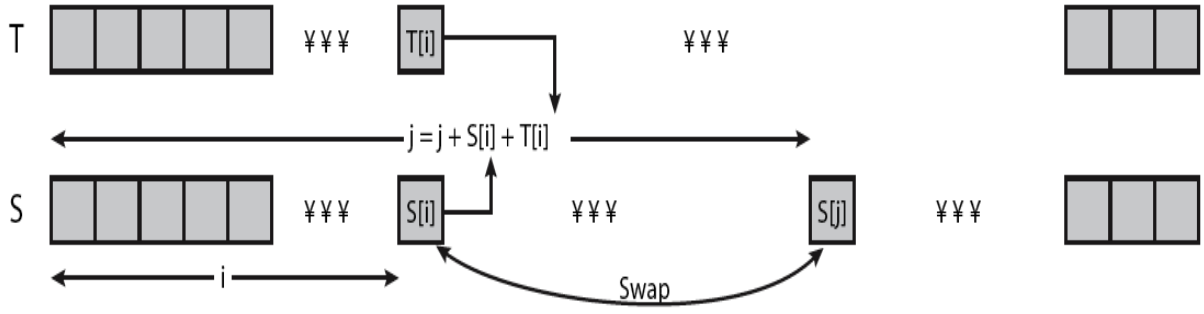
t = (S[i] + S[j]) (mod 256)

$C_i = M_i$ XOR S[t]

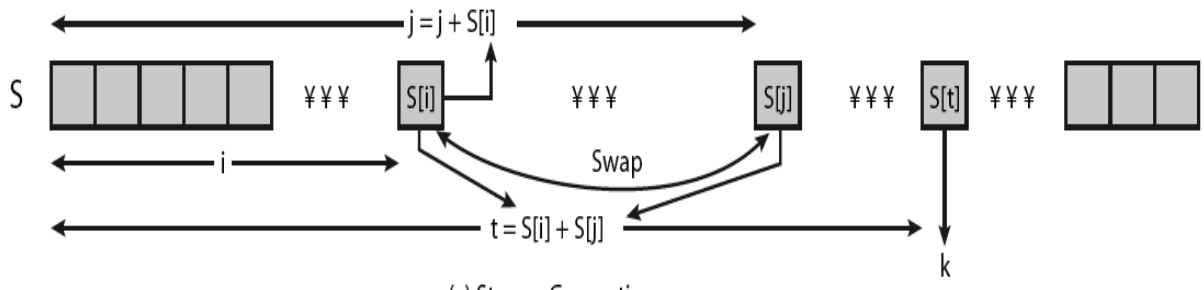
Overview of RC4:



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream Generation

RC4 Security:

- ✓ A number of papers have been published analyzing methods of attacking RC4, but none of these approaches is practical against RC4 with a reasonable key length, such as 128 bits.
- ✓ A more serious problem occurs in its use in the WEP protocol, not with RC4 itself but the way in which keys are generated for use as input to RC4.
- ✓ Currently RC4 it's regarded as quite secure, if used correctly, with a sufficiently large key.

KEY DISTRIBUTION:

- ➢ Symmetric schemes require both parties to share a common secret key
- ➢ Issue is how to securely distribute this key
- ➢ Often secure system failure due to a break in the key distribution scheme

Given parties A and B have various key distribution alternatives:

1. A can select key and physically deliver to B
2. Third party can select & deliver key to A & B
3. if A & B have communicated previously can use previous key to encrypt a new key
4. if A & B have secure communications with a third party C, C can relay key between A & B

The strength of any cryptographic system thus depends on the key distribution technique.

For two parties A and B, key distribution can be achieved in a number of ways:

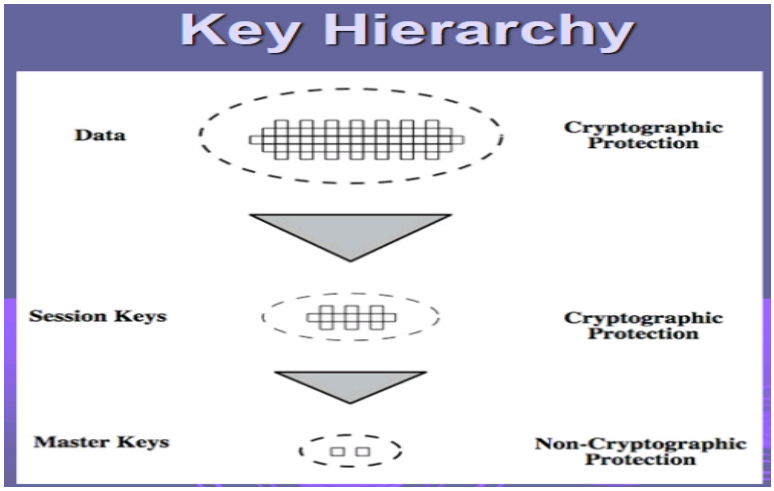
- Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows.
- 3 is mostly based on 1 or 2 occurring first.
- A third party, whom all parties trust, can be used as a **trusted intermediary** to mediate the establishment of secure communications between them (4).
- Must trust intermediary not to abuse the knowledge of all session keys. As number of parties grow, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.

KEY HIERARCHY:

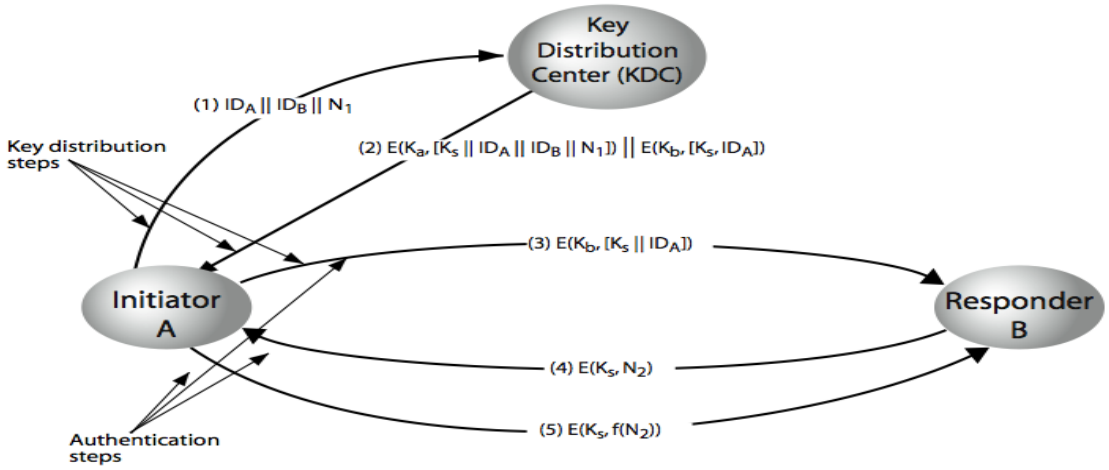
The use of a key distribution center is based on the use of a hierarchy of keys.

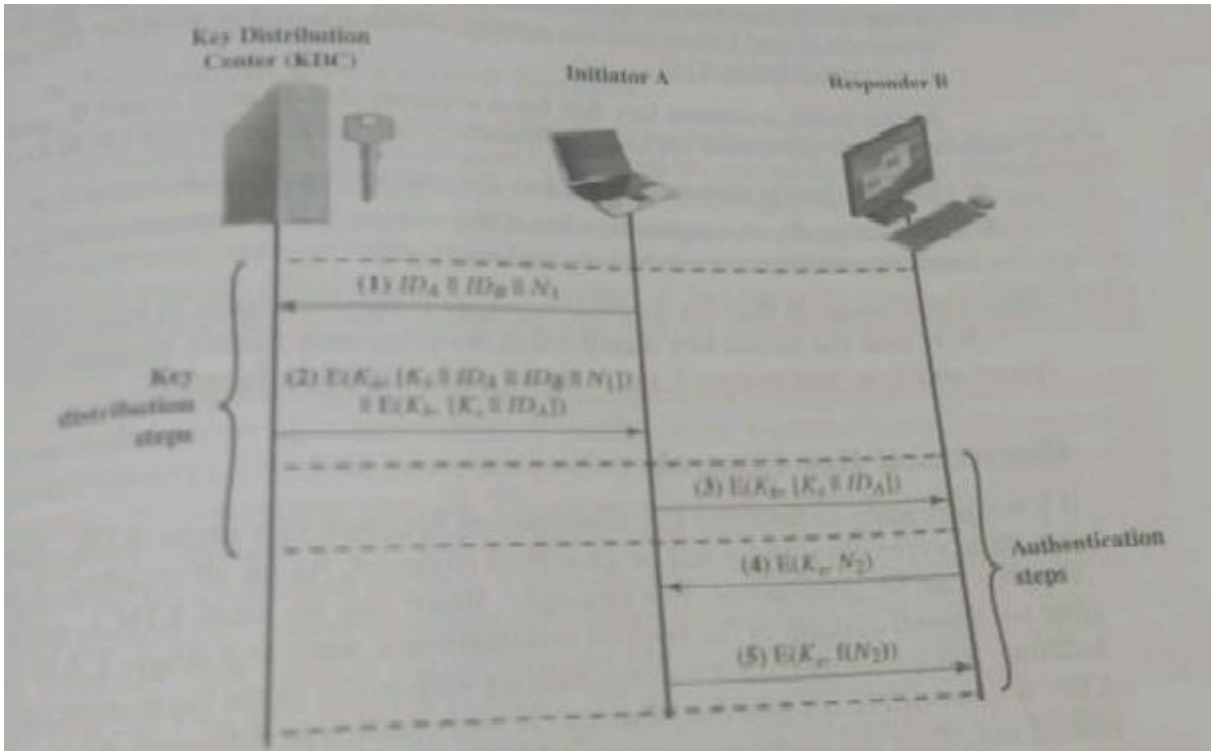
At a minimum, two levels of keys are used:

- A session key, used for the duration of a logical connection;
- A master key shared by the key distribution center and an end system or user and used to encrypt the session key.



Key Distribution Scenario:"Key Distribution Center" (KDC) which shares a unique key with each party (user)





The major issues associated with the use of Key Distribution Centers (KDC's):

- **Hierarchies of KDC's** required for large networks, but must trust each other
  Significant of hierarchical key control. (Nov / Dec 2017)
  - There can be local KDC(Key Distribution Center) responsible for small domain in the large networks.
  - When the two principals are in the same domain the local KDC does the key distribution.
  - When the two principals are in different domain, the local KDC communicates to the global KDC.
  - The key selection can be done by anyone KDC. The number of layers depend upon the network size.

- **session key lifetimes** should be limited for greater security
- use of **automatic key distribution** on behalf of users, but must trust system
- use of **decentralized key distribution**
- **Controlling key usage**

UNIT III PUBLIC KEY CRYPTOGRAPHY

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY: Primes – Primality Testing – Factorization – Euler's totient function, Fermat's and Euler's Theorem – Chinese Remainder Theorem – Exponentiation and logarithm

ASYMMETRIC KEY CIPHERS: RSA cryptosystem – Key distribution – Key management – Diffie Hellman key exchange -ElGamal cryptosystem – Elliptic curve arithmetic-Elliptic curve cryptography.

MATHEMATICS OF ASYMMETRIC KEY CRYPTOGRAPHY

PRIME NUMBER

An integer $p > 1$ is a prime number if and only if its only divisors are ±1 and ±p. Any integer $a > 1$ can be factored in a unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$$

where $p_1 < p_2 < \ldots < p_t$ are prime numbers and where each a_i is a positive integer.

Eg, $91 = 7 * 13$
$3600 = 2^4 * 3^2 * 5^2$
$11011 = 7 * 11^2 * 13$

If P is the set of all prime numbers, then any positive integer a can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \quad \text{where each } a_p \geq 0$$

It is easy to determine the greatest common divisor of two positive integers if we express each integer as the product of primes

Eg $300 = 2^2 * 3^1 * 5^2$
$18 = 2^1 * 3^2$
$\gcd(18, 300) = 2^1 * 3^1 * 5^0 = 6$

The following relationship always holds: If $k = \gcd(a, b)$, then $k_p = \min(a_p, b_p)$ for all p.

TESTING FOR PRIMALITY

For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random. Thus, we are faced with the task of determining whether a given large number is prime. There is no simple yet efficient means of accomplishing this task.

Miller-Rabin Algorithm

The algorithm due to Miller and Rabin [MILL75, RABI80] is typically used to test a large number for primality.

TEST (*n*)

1. Find integers k, q, with $k > 0$, q odd, so that $(n - 1 = 2^k q)$;
2. Select a random integer a, $1 < a < n - 1$;
3. if $a^q \bmod n = 1$ **then** return("inconclusive");
4. for $j = 0$ **to** $k - 1$ **do**
5. if $a^{2^j q} \bmod n = n - 1$ **then** return("inconclusive");
6. return("composite");

Example 1: Let us apply the test to the prime number $n = 29$.
$(n - 1) = 28 = 2^2(7) = 2^k q$.
First, let us try $a = 10$.
Compute $10^7 \bmod 29 = 17$,
$(10^7)^2 \bmod 29 = 28$, and the test returns inconclusive.
So n is prime number.

FERMAT AND EULER'S THEOREM

Two theorems that play important roles in public-key cryptography are Fermat's theorem and Euler's theorem

Fermat's Theorem (also called as Fermat's little Theorem)

Definition:

If P is prime and a is a positive integer not divisible by p, then $\mathbf{a^{p-1} \equiv 1 \pmod p}$

Fermat's little theorem is the basis for the Fermat primality test and is one of the fundamental results of elementary number theory.

This theorem is useful in generating public key in RSA and Primality testing

Proof:

1) Consider the set of positive integers less than *p:*{1,2,3..p-1}
2) Multiply each element by *a, modulo p* to get the set

   X ={ a mod p,2a mod p….(p-1) a mod p}.

   - None of the elements of X is equal to zero because *p* does not divide *a*.
   - No two of the integers in X are equal.
   - We know that (p-1) elements of X are all positive integers with no two elements are equal. So, we can conclude X consists of the set of integers {1,2,……..,p-1} in some order

3) Multiplying the numbers in both sets (p and X) and taking the result mod *p* yields.

$$a * 2a * \ldots * (p-1)a \equiv [(1*2*\ldots*(p-1)] \pmod p$$
$$\{1 * 2 * \ldots * (p-1)\}\ a^{p-1} \equiv [(1*2*\ldots*(p-1)] \pmod p$$
$$(p-1)!\ a^{p-1} \equiv (p-1)! \pmod p$$
$$\mathbf{a^{p-1} \equiv 1 \pmod p}$$

Example

$a = 7, p = 19$ $\quad a^{p-1} \equiv 1 \pmod p.$

- $7^2 = 49 \equiv 11 \pmod{19}$
- $7^4 = 7^2 \times 7^2 = 121 \equiv 7 \pmod{19}$
- $7^8 = 7^4 \times 7^4 = 7 \times 7 = 49 \equiv 11 \pmod{19}$
- $7^{16} \equiv 7^8 \times 7^8 = 11 \times 11 = 121 \equiv 7 \pmod{19}$
- $a^{p-1} = 7^{18} = 7^{16} * 7^2 \equiv 7 * 11 \equiv 1 \pmod{19}$

An alternative form of Fermat's theorem:

If p is prime and a is a positive integer, $\mathbf{a^p \equiv a \pmod p}$

Eg: a=3, p=5
$a^p = 3^5 = 243 \equiv 3 \pmod 5 = a \pmod p$
$a^p \equiv a \pmod p$

Euler's Totient function

Euler's totient function written as $\emptyset(n)$, (called as phi) is defined as the number of positive integers less than n and relatively prime (Co-Prime) to ***n***.

The Properties are as follows

1) ø(1) = 1
2) ø(p)=p-1 for p (p prime)
3) ø(p.q)=(p-1)x(q-1) for p.q (p,q prime)

Suppose that we have two prime numbers *p* and *q*, with *p* not equal to *q*. Then we can show that

n=pq.

ø(n)= ø(pq)= ø(p)* ø(q)=(p-1)*(q-1)

Examples:

1) ø(37) = 36 {ø(p)=p-1 for p (p prime)]
2) ø(21) = ø(3)* ø(7)= (3–1)x(7–1) = 2x6 = 12 where the 12 integers are {1,2,4,5,8,10,11,13,16,17, 19, 20}

[ø(p.q)=(p-1)x(q-1) for p.q (p,q prime)]

Sample Examples

1. What is the value of Φ(13)?
   Because 13 is a prime, Φ (13) = (13 −1) = 12.
2. What is the value of Φ (10)?
   We can use the third rule: Φ (10) = Φ (2) × Φ (5) = 1 × 4 = 4, because 2 and 5 are primes.
3. What is the number of elements in Z14*?
   The answer is Φ (14) = Φ (7) × Φ (2) = 6 × 1 = 6. The members are 1, 3, 5, 9, 11, and 13.

Euler's theorem

Euler's theorem states that for every *a* and *n* that are relatively prime:

$$\mathbf{a^{ø(n)} \equiv 1 \pmod n}$$

Proof:

The above equation is true, if n is prime, because in that case *ø(n)=(n-1)* and Fermat's theorem holds. However it holds for any integer *n*.

1)Recall that ø(n)is the number of positive integers less than n that are relatively prime to *n*. consider the set of such integers, labeled as follows:

$$R=\{x_1, x_2 \ldots . x_{ø(n)}\}$$

That is, each element x_i of R is a unique positive integer less than *n* with $\gcd(x_i, n)=1$.

2)Now multiply each element by *a modulo n:*

$$S=\{(ax_1 \bmod n), (ax_2 \bmod n), \ldots . (ax_{ø(n)}) \bmod n)\}$$

3) The set S is a permutation of R, by the following reasons:
1.Because *a* is relatively prime to *n* and x_i is relatively prime to *n*,ax_i must also be relatively prime to *n*. thus all the members of S are integers that are less than n and that are relatively prime to n.
2. There are no duplicates in S. if $ax_i \bmod n = ax_i \bmod n$, then $x_i = x_j$

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i \right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

An alternative form of the theorem is also useful:

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

$$a = 3; n = 10; \phi(10) = 4 \quad a^{\phi(n)} = 3^4 = 81 = 1 \pmod{10} = 1 \pmod{n}$$
$$a = 2; n = 11; \phi(11) = 10 \quad a^{\phi(n)} = 2^{10} = 1024 = 1 \pmod{11} = 1 \pmod{n}$$

THE CHINESE REMAINDER THEOREM

The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime, as shown below:

Let $m_1, m_2 \ldots\ldots m_k$ be integers with $\gcd(m_i, m_j) = 1$, whenever $i \neq j$. Let $a_1, a_2 \ldots\ldots a_k$ be integers, there exists exactly one solution x (mod $m_1, m_2 \ldots\ldots m_k$) to the simultaneous congruences

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$x \equiv a_k \pmod{m_k}$$

If $n_1,n_2,..,n_k$ are positive integers that are pairwise co-prime and $a_1,a_2,\ldots,a_k$ are any integers, then CRT is used to find the values of x that solves the following congruence simultaneously.

Value of $x=(a_1m_1y_1+a_2m_2y_2+\ldots+a_km_ky_k) \bmod M$
Where $M=n_1n_2n_3..n_k$
$m_i=M/n_i$
$m_iy_i=1 \bmod n_i$

Example 1:
Find the solution to the simultaneous equations: $x \equiv 1 \pmod 5$, $x \equiv 2 \pmod 6$, $x \equiv 3 \pmod 7$.
Solution
$M=n_1n_2n_3$
$M=5*6*7=210$

$m_i = M/n_i$

m1=210/5=42

m2=210/6=35

m3=210/7=30

$m_i y_i = 1 \bmod n_i$

$42y_1 = 1 \bmod 5$

$y_1 = 2$

$35y_2 = 1 \bmod 6$

$y_2 = 5$

$30y_3 = 1 \bmod 7$

$y_3 = 2$

$x = (a_1 m_1 y_1 + a_2 m_2 y_2 + a_3 m_3 y_3) \bmod M$

$= ((1*42*2)+(2*35*5)+(3*30*3)) \bmod 210$

$= 193$

Example 2:

Find an integer that has a remainder of 3 when divided by 7 and 13, but is divisible by 12.
Solution: This is a CRT problem. We can form three equations and solve them to find the value of x.

$$x = 3 \bmod 7$$
$$x = 3 \bmod 13$$
$$x = 0 \bmod 12$$

If we follow the four steps, we find x = 276. We can check that 276 = 3 mod 7, 276 = 3 mod 13 and 276 is divisible by 12 (the quotient is 23 and the remainder is zero).

Example 3:

A bag has contained number of pens if you take out 3 pens at a time 2 pens are left. If you take out 4 pens at a time 1 pen is left and if you take out 5 pens at a time 3 pens are left in the bag. What is the number of pens in the bag.

$x \equiv 2 \bmod 3$

$x \equiv 1 \bmod 4$

$x \equiv 3 \bmod 5$

$a_1 = 2$

$a_2 = 1$

$a_3 = 3$

$n_1 = 3$

$n_2 = 4$

$n_3 = 5$

$M = n_1 n_2 n_3$

$M = 3*4*5 = 60$

$m_i=M/n_i$
m1=60/3=20
m2=60/4=15
m3=60/5=12

$m_iy_i=1 \bmod n_i$
$20y_1=1 \bmod 3$
$Y_1=2 \bmod 3$

$15y_2= 1 \bmod 4$
$y_2=3 \bmod 4$

$12y_3=1 \bmod 5$
$y_3=3 \bmod 5$

$x=(a_1m_1y_1+a_2m_2y_2+ a_3m_3y_3) \bmod M$
=((2*20*2)+(1*15*3)+(3*12*3)) mod 60
=233 mod 60
=53

DISCRETE LOGARITHMS.

Discrete logarithms are fundamental to a number of public-key algorithms. Discrete logarithms are analogous to ordinary logarithms but are defined using modular arithmetic.

Discrete logarithms are fundamental to a number of public-key algorithms, including Diffie-Hellman key exchange and the digital signature algorithm (DSA)

The Powers of an Integer, Modulo *n*

Recall from Euler's theorem that, for every and that are relatively prime,

$$a^{\phi(n)} = 1 \pmod{n}$$

Where $\phi(n)$, Euler's totient function, is the number of positive integers less than and relatively prime to . Now consider the more general expression:

$$a^m = 1 \pmod{n}$$

If a and n are relatively prime, then there is at least one integer m that satisfies Equation, namely M= $\phi(n)$, . The least positive exponent m for which Equation holds is referred to in several ways:

- The order of a(mod n)
- The exponent to which a belongs (mod n)
- The length of the period generated by a

The highest possible exponent to which a number can belong (mod n) is $\phi(n)$. If a number is of this order, it is referred to as a **primitive root** of n .The importance of this notion is that if is a primitive root of , then its powers

$$a, a^2, \ldots, a^{\phi(n)}$$

Logarithms for Modular Arithmetic

A primitive root of a prime number p is 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^{p-1} mod p

are distinct and consist of the integers from 1 through p - 1 in some permutation.

For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$b \equiv a^i \pmod p$ where 0 … i … (p - 1)

The exponent i is referred to as the discrete logarithm of b for the base a, mod p.

We denote this value as dloga,p(b)

Note the following:

dlog$_{a,p}(a)$ = 1 because $a^1 \bmod p = a$

dlog$_{a,p}(1)$ = 0 because $a^0 \bmod p = 1 \bmod p = 1$

Calculation of Discrete Logarithms

Consider the equation

$$y = gx \bmod p$$

Given g, x , and p , it is a straightforward matter to calculate y. At the worst, we must perform repeated multiplications, and algorithms exist for achieving greater efficiency.

PUBLIC KEY CRYPTOGRAPHY

Introduction to Public key Cryptography:

- Public key cryptography also called as **asymmetric cryptography**.
- It was invented by whitfield **Diffie** and Martin **Hellman** in 1976. Sometimes this cryptography also called as **Diffie-Helman Encryption**.
- Public key algorithms are based on mathematical problems which admit no efficient solution that are inherent in certain integer factorization, discrete logarithm and Elliptic curve relations.

Public key Cryptosystem Principles:

- The concept of public key cryptography in invented for two most difficult problems of Symmetric key encryption.
  - *The Key Exchange Problem*
  - *The Trust Problem*

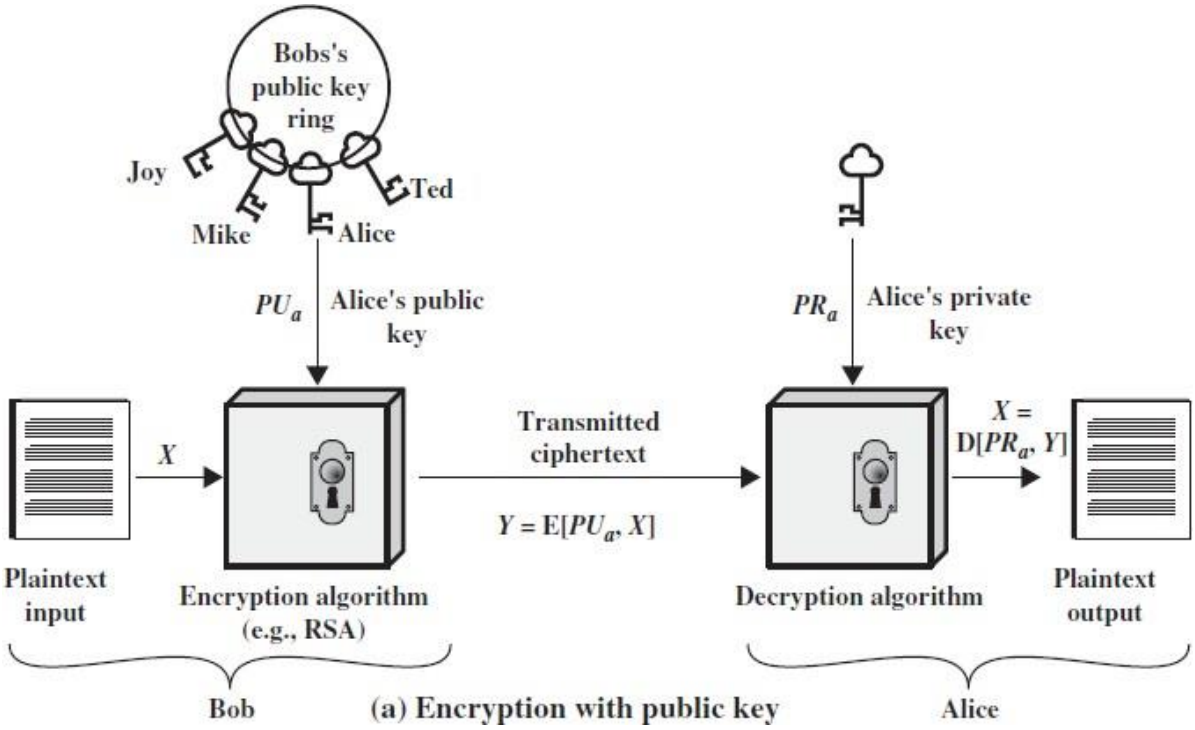
  The Key Exchange Problem: *The key exchange problem arises from the fact that communicating parties must somehow share a secret key before any secure communication can be initiated, and both parties must then ensure that the key remains secret. Of course, direct key exchange is not always feasible due to risk, inconvenience, and cost factors.*

  The Trust Problem: *Ensuring the integrity of received data and verifying the identity of the source of that data can be very important. Means in the symmetric key cryptography system, receiver doesn't know whether the message is coming for particular sender.*

- This public key cryptosystem uses two keys as pair for encryption of plain text and Decryption of cipher text.
- These two keys are names as "**Public key**" and "**Private key**". The private key is kept secret whereas public key is distributed widely.
- A message or text data which is encrypted with the public key can be decrypted only with the corresponding private-key
- This two key system very useful in the areas of confidentiality (secure) and authentication

A **public-key encryption** scheme has six ingredients		
1	**Plaintext**	This is the readable message or data that is fed into the algorithm as input.
2	**Encryption algorithm**	The encryption algorithm performs various transformations on the plaintext.
3	**Public key**	This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input
4	**Private Key**	
5	**Ciphertext**	This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
6	**Decryption algorithm**	This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Public key cryptography for providing confidentiality (secrecy)



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. So above fig states that 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.

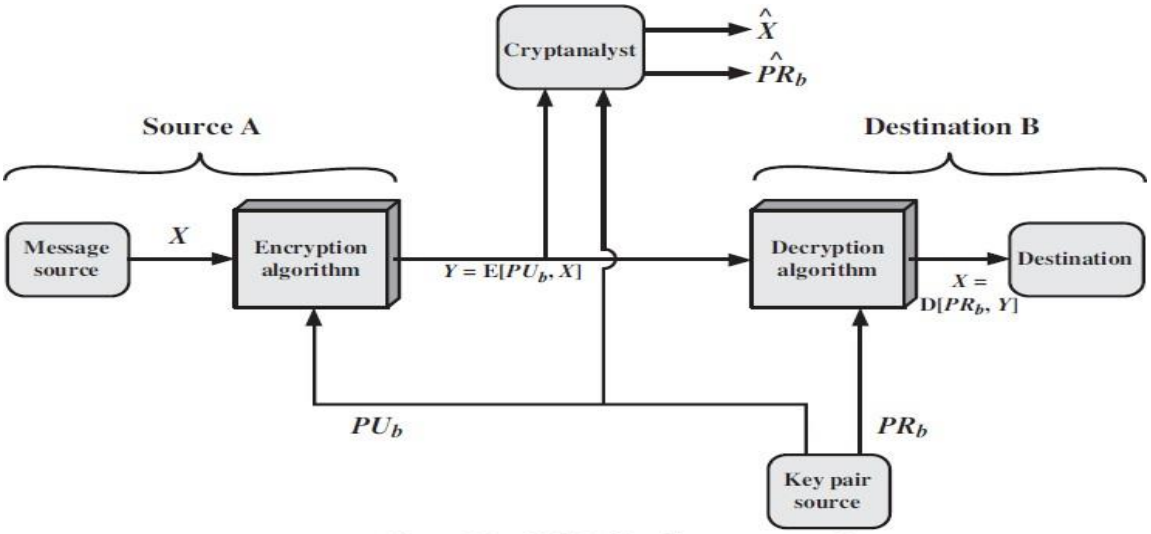


Figure 9.2 Public-Key Cryptosystem: Secrecy

There is some source A that produces a message in plaintext $\boldsymbol{X = [X1, X2, \ldots, XM]}$.

The $\boldsymbol{M}$ elements of $\boldsymbol{X}$ are letters in some finite alphabet. The message is intended for destination **B**. B generates a related pair of keys: a public key, $\boldsymbol{PU_b}$, and a private key, $\boldsymbol{PR_b}$.

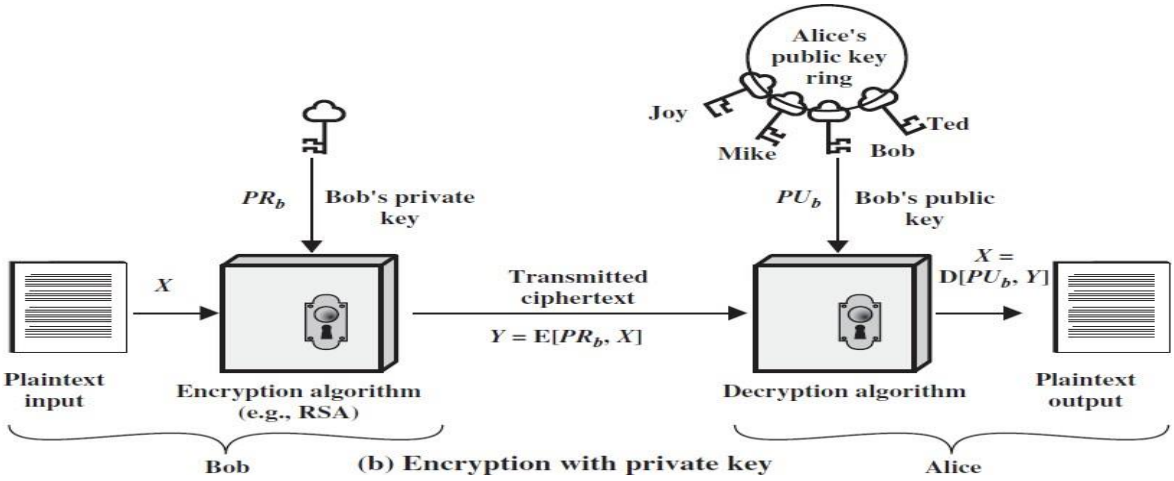
$\boldsymbol{PR_b}$ is known only to B, whereas $\boldsymbol{PU_b}$ is publicly available and therefore accessible by A. With the message X and the encryption key $\boldsymbol{PU_b}$ as input, A forms the ciphertext $Y = [Y1, Y2, \ldots, Y_N]$:

$$Y = \mathrm{E}(PU_b, X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = \mathrm{D}(PR_b, Y)$$

Public key cryptography for proving Authentication:



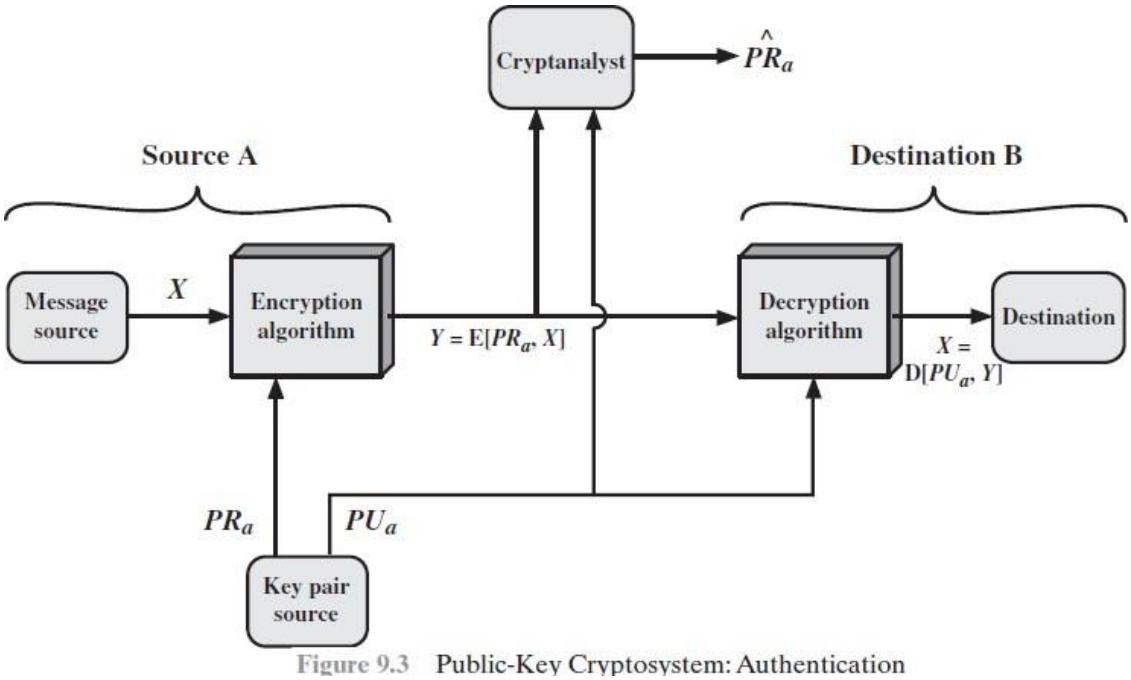


Figure 9.3 Public-Key Cryptosystem: Authentication

The above diagrams show the use of public-key encryption to provide authentication:

$$Y = \mathrm{E}(PR_a, X)$$

$$X = \mathrm{D}(PU_a, Y)$$

- In this case,A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature.**
- It is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

Public key cryptography for both authentication and confidentiality (Secrecy)

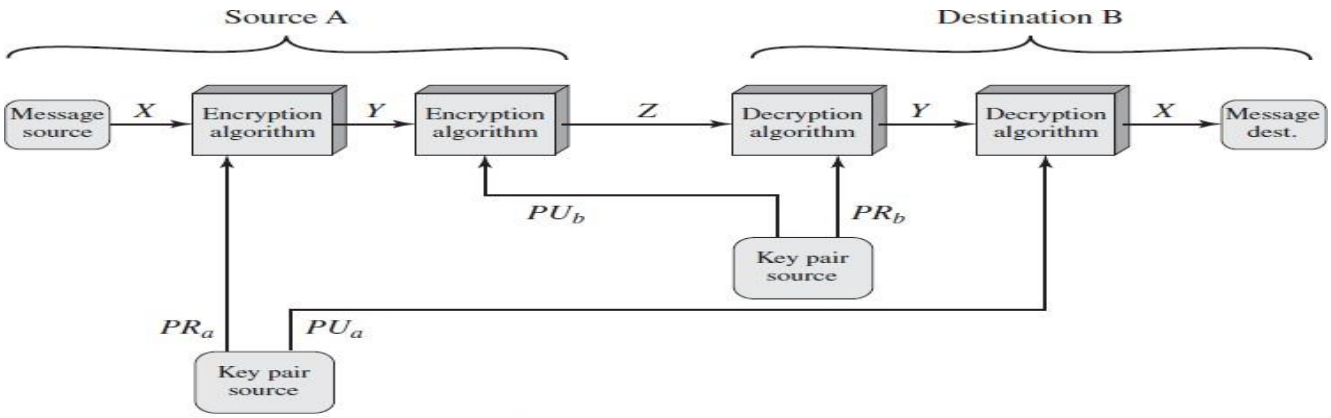


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (above figure):

$$Z = \mathrm{E}(PU_b, \mathrm{E}(PR_a, X))$$

$$X = \mathrm{D}(PU_a, \mathrm{D}(PR_b, Z))$$

In this case, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided.

Applications for Public-Key Cryptosystems

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.

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

RSA

- It is the most common public key algorithm.
- This RSA name is get from its inventors first letter (Rivest (R), Shamir (S) and Adleman (A)) in the year 1977.
- The RSA scheme is a block cipher in which the plaintext & 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}

Description of the Algorithm:

- RSA algorithm uses an expression with exponentials.

- In RSA 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 **$\log_2(n)$**
- **RSA** uses two exponents 'e' and 'd' where e is public and d is private.
- Encryption and decryption are of following form, for some PlainText 'M' and CipherText block 'C'

$$C = M^e \bmod n$$

$$M = C^d \bmod n$$

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

- Both sender and receiver must know the value of n.
- The sender knows the value of 'e' & only the reviver knows the value of 'd' thus this is a public key encryption algorithm with a

  Public key PU={e, n}

  Private key PR={d, n}

Requirements:

The RSA 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 " $\mathbf{M^{ed} \bmod n = M}$ " for all M<n
2. It is relatively easy to calculate " $\mathbf{M^e \bmod n}$ " and " $\mathbf{C^d \bmod n}$ "for M<n
3. It is infeasible to determine "d" given 'e' & 'n'. The " $\mathbf{M^{ed} \bmod n = M}$ " relationship holds if 'e' & 'd' are multiplicative inverses modulo Ø(n).

   Ø(n) is Euler Totient function

   For p,q primes where p*q and p≠q.

   Ø(n)= Ø(pq)=(p-1)(q-1)

Then the relation between 'e' & 'd' can be expressed as " $\boxed{ed \bmod \emptyset(n)=1}$ "
this is equivalent to saying

$$ed \equiv 1 \bmod \emptyset(n)$$

$$d \equiv e^{-1} \bmod \emptyset(n)$$

That is 'e' and 'd' are multiplicative inverses mod Ø(n).

Note: according to the rules of modular arithmetic, this is true only if 'd' (and 'e') is relatively prime to Ø(n).

Equivalently **gcd(Ø(n), d)=1.**

Steps of RSA algorithm:

Step 1 Select 2 prime numbers p & q

Step 2 Calculate n=pq

Step 3Calculate Ø(n)=(p-1)(q-1)

Step 4 Select or find integer e (public key) which is relatively prime to Ø(n) i.e., e with gcd (Ø(n), e)=1 where 1<e< Ø(n).

Step 5 Calculate "d" (private key) by using following condition. $ed \equiv 1 \bmod \emptyset(n)$

d< Ø(n).

Step 6 Perform encryption by using $C = M^e \bmod n$

Step 7 perform Decryption by using $M = C^d \bmod n$

Example:

1. Select two prime numbers, $p = 17$ and $q = 11$.

2. Calculate $n = pq = 17 \times 11 = 187$.

3. Calculate $\emptyset(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.

4. Select e such that e is relatively prime to $\emptyset(n) = 160$ and less than $\emptyset(n)$; we choose $e = 7$.

5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 * 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm

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 $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

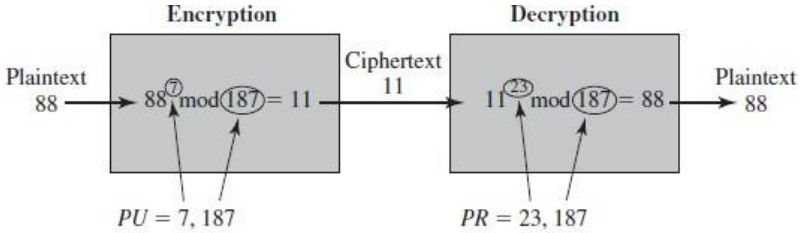


Figure 9.6 Example of RSA Algorithm

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

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

RSA Attacks

There are four possible approaches to attack the RSA:

- **Brute force:** This involves trying all possible private keys. The defence against this attack is the use of large key space.
- **Mathematical attacks**: There are several approaches, all equivalent in effort to factoring the product of two primes. Three approaches that could be identified of this type are:
    - Factor n into its two prime factors which enables calculation of $\phi(n) = (p - 1) \times (q - 1)$, which in turn enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
    - Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
    - Determine d directly, without first determining $\phi(n)$.
- **Timing attacks**: These depend on the running time of the decryption algorithm. This attack is alarming for two reasons namely it comes from a completely unexpected direction, and it is a cipher text-only attack. Modular exponentiation algorithm that is accomplished bit by bit, with one modular multiplication performed every iteration and an additional modular multiplication performed for each 1 bit can be used to perform this attack. Simple counter measures could be used to overcome the timing attack. They are
    - ***Constant exponentiation time***: Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
    - ***Random delay***: Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. Kocher points out that if defenders don't add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays.
    - ***Blinding***: Multiply the cipher text by a random number before performing exponentiation. This process prevents the attacker from knowing what cipher text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.
- **Chosen cipher text attacks (CCAs)**: This type of attack exploits properties of the RSA algorithm. It is defined as an attack in which the adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's

private key. Thus, the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key. Clearly, this provides the adversary with no new information. Instead, the adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis.

To overcome this simple attack, practical RSA-based cryptosystems randomly pad the plaintext prior to encryption. More sophisticated CCAs are possible and simple padding with a random value is insufficient to provide the desired security. To counter such attacks modifying the plaintext using a procedure known as optimal asymmetric encryption padding will help.

Diffie-Hellman key exchange is the first published public key algorithm, also known as exponential key agreement. And it is based on mathematical principles. The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.This algorithm itself is limited to exchange of the keys. Security of algorithm depends on computing discrete logarithms values.

KEY MANAGEMENT

There are actually two distinct aspects to the use of public-key cryptography:

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

1. Distribution of Public Keys

There are four different schemes

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

A.Public announcement

Any participant can send his or her public key to any other participant or **broadcast** the key to the community. Uncontrolled Public-Key Distribution

Limitation : Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Authentication is needed to avoid this problem.

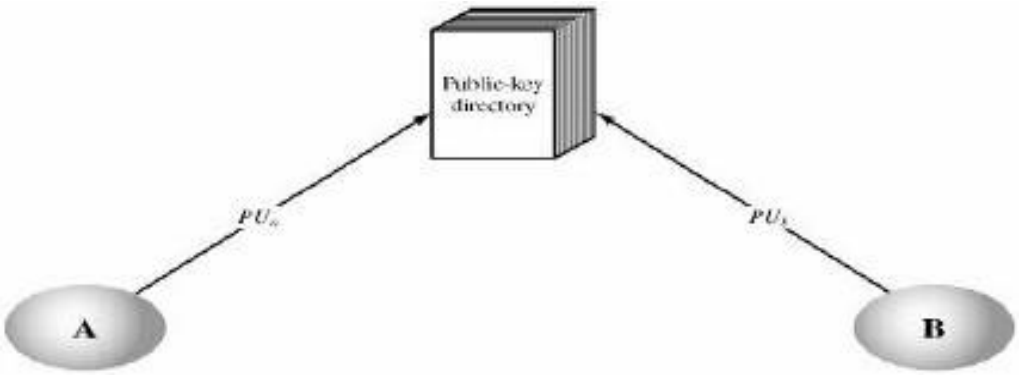
B. Publicly Available Directory

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.

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

ii)Each participant registers a public key with the directory authority.

iii) Participants could also access the directory electronically.



Limitation :

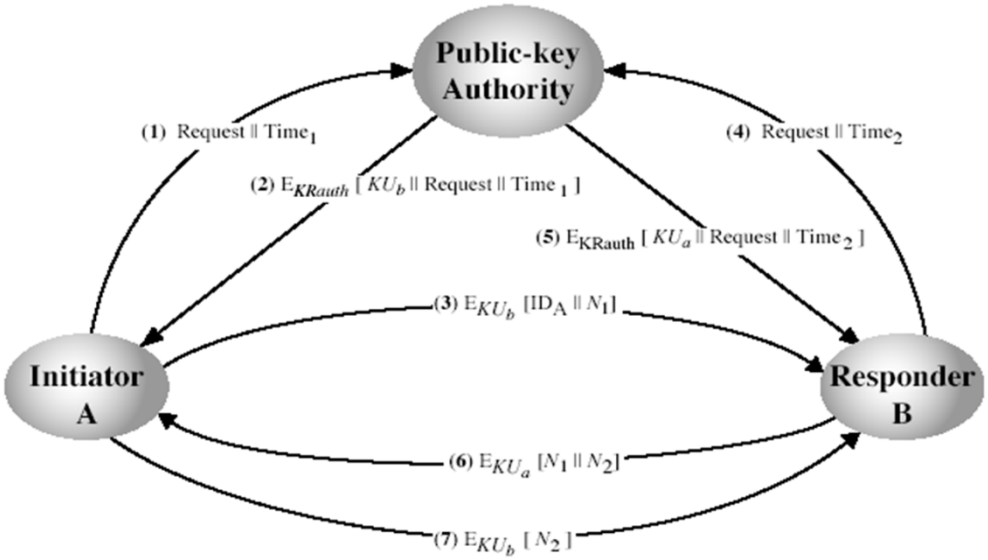
An Adversary may impersonate by stealing the private key of public key directory and falsely send the public key details.

An attacker may attack the records stored in the directory.

C. 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.

Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.



i)A sends a time stamped message to the public-key authority containing a request for the current public key of B.

ii) The authority responds with a message that is encrypted using the authority's private key, PRauth. Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured

that the message originated with the authority. The message includes the following:

- B's public key, PUb which A can use to encrypt messages destined for B
- The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
- The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key

iii) A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely.

iv) B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

v) At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange.

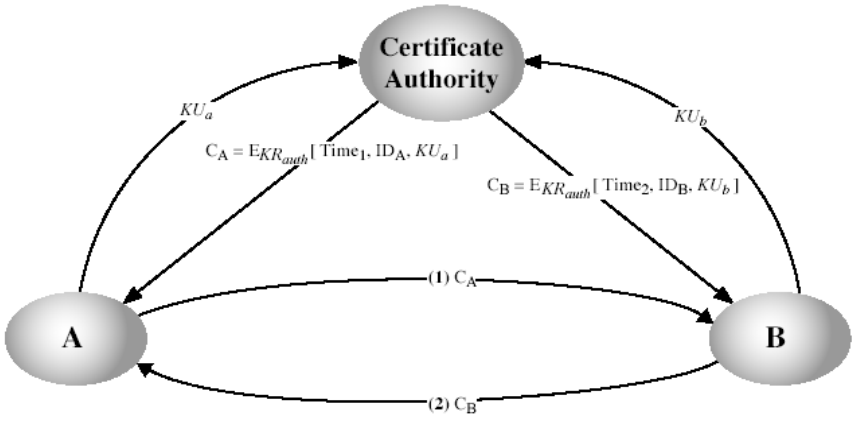
v) B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2) Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.

vi) A returns N2, encrypted using B's public key, to assure B that its correspondent is A.

Limitations : Bottleneck may occur in public authority. Tampering of records stored by the authority may take place.

D. Public key certificate

A certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party.



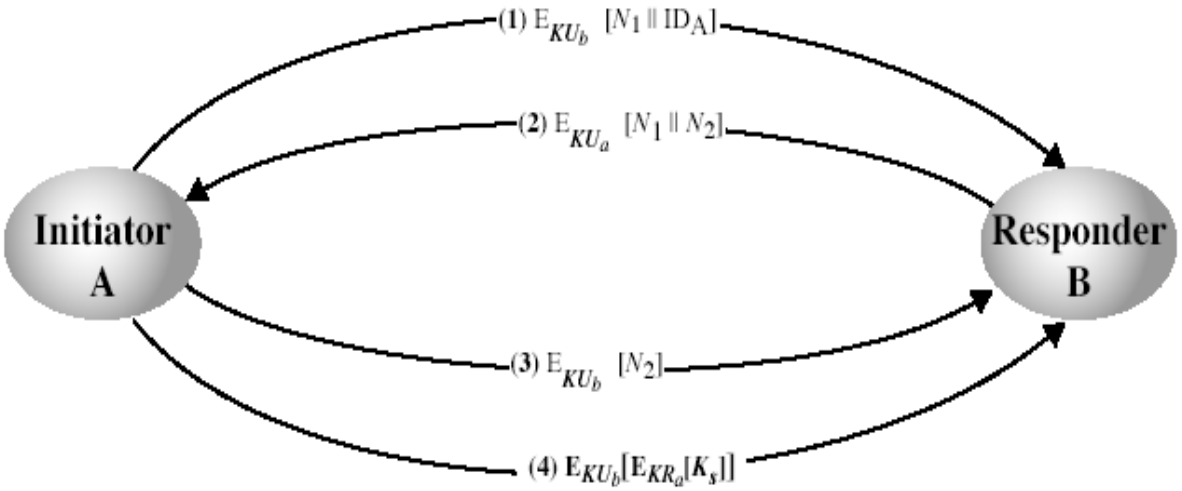
Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community.

A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate.

2. Secret Key Distribution with Confidentiality and Authentication

i) A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely.

ii) B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2)



iv) A returns *N*2 encrypted using B's public key, to assure B that its correspondent is A.
A selects a secret key *Ks* and sends *M* = E(*PUb*, E(*PRa*, *Ks*)) 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.

v) B computes D(*PUa*, D(*PRb*, *M*)) to recover the secret key.

Elagamal Cryptographic system

- Public-key cryptosystem related to D-H
- uses exponentiation in a finite field
- with security based difficulty of computing discrete logarithms, as in D-H
- Used in number of standards including DSS (Digital Signature Standard) and S/MIME e-mail standard

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

How K is recovered by the decryption process:

$K = (Y_A)^k \bmod q$ — K is defined during the encryption process

$K = (\alpha^{X_A} \bmod q)^k \bmod q$ — substitute using $Y_A = \alpha^{X_A} \bmod q$

$K = \alpha^{kX_A} \bmod q$ — by the rules of modular arithmetic

$K = (C_1)^{X_A} \bmod q$ — substitute using $C_1 = \alpha^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$

$$(C_2 K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

Example:

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$.
3. Alice's private key is 5 and Alice's public key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then:

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = \text{KM} \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

DIFFIE- HELLMAN KEY EXCHANGE

Algorithm for Diffie-Hellman Key Exchange:

Step 1 two public known numbers q, α

q Prime number

α primitive root of q and α< q.

Step 2 if A & B users wish to exchange a key

a) User A select a random integer X_A<q and computes $Y_A = \alpha^{X_A} \bmod q$

b) User B independently select a random integer X_B <q and computes $Y_B = \alpha^{X_B} \bmod q$

c) Each side keeps the X value private and Makes the Y value available publicly to the outer side.

Step 3 User A Computes the key as $\boxed{K = (Y_B)^{X_A} \bmod q}$

User B Computes the key as $\boxed{K = (Y_A)^{X_B} \bmod q}$

Step 4 two calculation produce identical results

$K = (Y_B)^{X_A} \bmod q$

$K = (\alpha^{X_B} \bmod q)^{X_A} \bmod q$ (We know that $Y_B = \alpha^{X_B} \bmod q$)

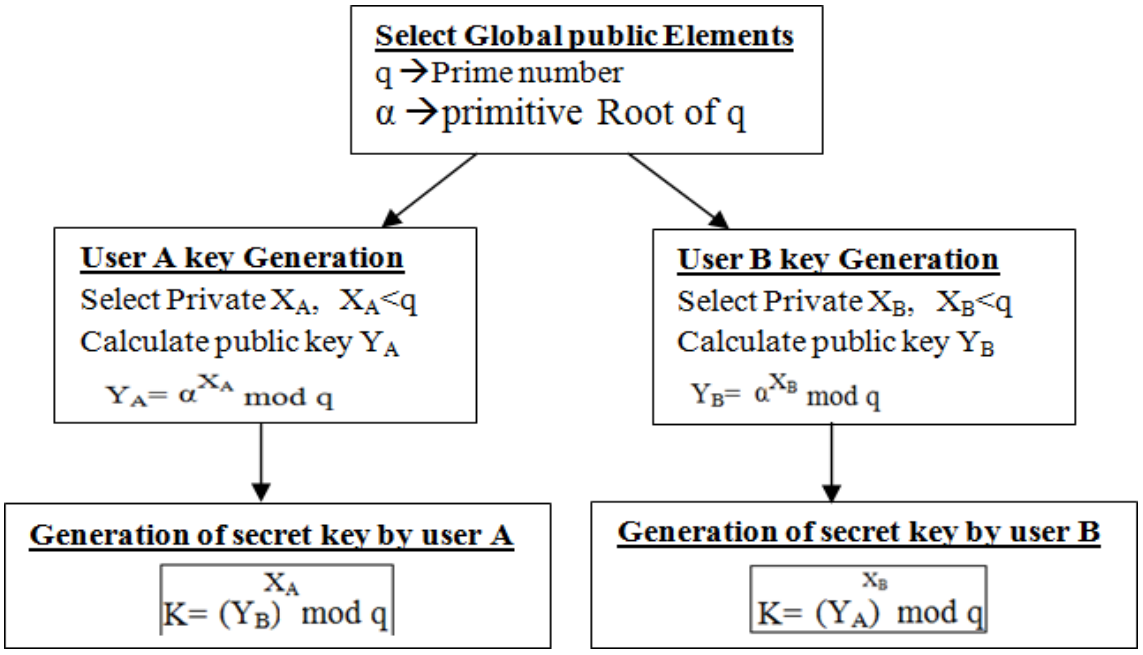
$= (\alpha^{X_B})^{X_A} \bmod q$

$= (\alpha^{X_A})^{X_B} \bmod q$

$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$

$= (Y_A)^{X_B} \bmod q$ (We know that $Y_A = \alpha^{X_A} \bmod q$)

The result is that the two sides have exchanged a secret key.



Example: 1

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 $\alpha = 3$. 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 $Y_B = 3^{233} \bmod 353 = 248$.

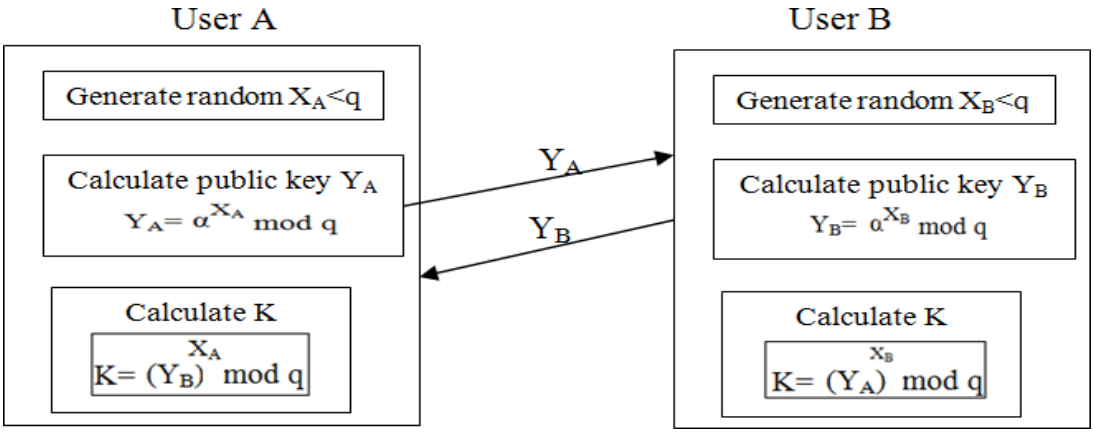
After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$



Example2:

Here is an example, taken from. Key exchange is based on the use of the prime number q=71 and a primitive root of 71, in this case $\alpha = 7$. A and B select private keys $X_A = 5$ and $X_B = 12$, respectively. Each computes its public key:

$$Y_A = 7^5 = 51 \bmod 71$$
$$Y_B = 7^{12} = 4 \bmod 71$$

After they exchange public keys, each can compute the common secret key:

$$K = (Y_B)^{X_A} \bmod 71 = 45 = 30 \bmod 71$$
$$K = (Y_A)^{X_B} \bmod 71 = 5112 = 30 \bmod 71$$

From {51, 4}, an attacker cannot easily compute 30.

Example 3:

User A and B exchange the key using Diffie-Hellman algorithm. Assume α=5 q=11 X_A=2 X_B=3. Find the value of Y_A, Y_B and k?

Soln:

$Y_A = \alpha^{X_A} \bmod q$
$= 25 \bmod 11$
$= 3$

$Y_B = \alpha^{X_B} \bmod q$
$= 125 \bmod 11$
$= 4$

$K = (Y_A)^{X_B} \bmod q$
$= 27 \bmod 11$
$= 5$

$K = (Y_B)^{X_A} \bmod q$
$= 16 \bmod 11$
$= 5$

MAN-IN-MIDDLE-ATTACK:

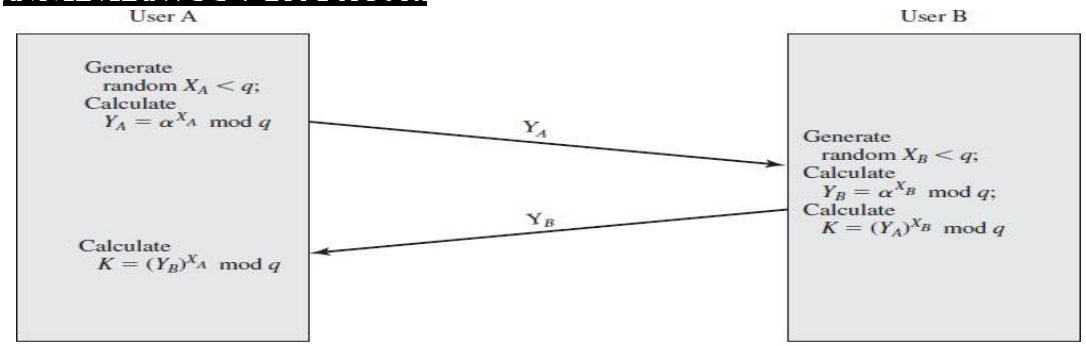


Figure 10.2 Diffie-Hellman Key Exchange

Definition: A man in the middle attack is a form of eavesdropping where communication between two users is monitored and modified by an unauthorized party.

Generally the attacker actively eavesdrops by intercepting (stoping) a public key message exchange.

The Diffie- Hellman key exchange is insecure against a "Man in the middle attack".

Suppose user 'A' & 'B' wish to exchange keys, and D is the adversary (opponent). The attack proceeds as follows.

1. 'D' prepares for the attack by generating two random private keys X_{D1} & X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. 'A' transmits 'Y_A' to 'B'
3. 'D' intercepts Y_A and transmits Y_{D1} to 'B'. and D also calculates $K2 = (Y_A)^{X_{D2}} \bmod q$.

4. 'B' receives Y_{D1} & calculate $K1 = (Y_{D1})^{X_B} \bmod q$.
5. 'B' transmits 'YB' to 'A"
6. 'D' intercepts 'YB' and transmits YD2 to 'A' and 'D' calculate K1 $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. A receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A} \bmod q$

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2. All future communication between Bob and Alice is compromised in the following way.

1. A sends an encrypted message *M*: E(*K*2, *M*) .
2. D intercepts the encrypted message and decrypts it to recover *M*.
3. D sends B $E(K1, M)$ or $E(K1, M')$, where M' is any message. In the first case, D simply wants to eavesdrop on the communication without altering it. In the second case, D wants to modify the message going to B

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates.

ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curve Arithmetic:

A major issue with the use of Public-Key Cryptography, is the size of numbers used, and hence keys being stored. Recently, an alternate approach has emerged, elliptic curve cryptography (ECC), which performs the computations using elliptic curve arithmetic instead of integer or polynomial arithmetic.

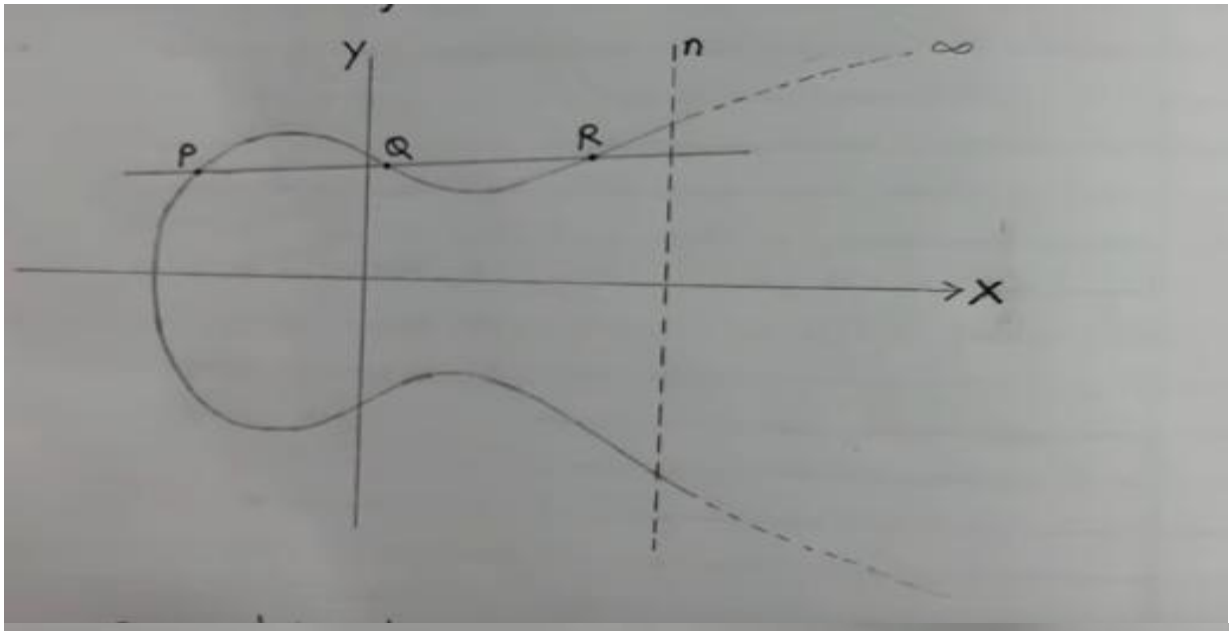
Majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials. It imposes a significant load in storing and processing keys and messages. An alternatives to use elliptic curves; it offers same security with smaller bit sizes.

Real Elliptic Curves

Elliptic Curve Cryptography (ECC)

- It is an asymmetric / public key cryptosystem
- It provides equal security with smaller key size as compared to non ECC algorithms
- It makes use of elliptic curves
- Elliptic curves are defined by some mathematical functions

Ex - $y^2 = x^3 + ax + b$



- symmetric to x-axis
- If we draw a line, it will touch a maximum of 3 points.

ECC Algorithm

ECC Key Exchange

Global Public Elements -

1) $E_q(a, b)$ - Elliptic curve with parameters a, b of q (prime number or an integer of the form 2^m)
2) G - Point on the elliptic curve

User A key generation

Select private key n_A $\quad n_A < n$

Calculate public key P_A $\quad P_A = n_A \times G$

User B key generation

Select private key n_B $\quad n_B < n$

Calculate public key P_B $\quad P_B = n_B \times G$

Calculation of secret key by user A

$$K = n_A \times P_B$$

Calculation of secret key by user B

$$K = n_B \times P_A$$

ECC Encryption

- Let the message be M
- First encode this message M into a point on elliptic curve
  Let this point be P_m

- For encryption, choose a random positive integer k
  The cipher point will be

$$C_m = \{kG, P_m + kP_B\}$$

This point will be sent to the receiver

Decryption

- For decryption, multiply x-coordinate with receiver's secret key

$$kG \times n_B$$

Then subtract $(kG \times n_B)$ from y-coordinate of cipher point

$$P_m + kP_B - (kG \times n_B)$$

we know that $P_B = n_B \times G$

$$\therefore P_m + kP_B - kP_B$$

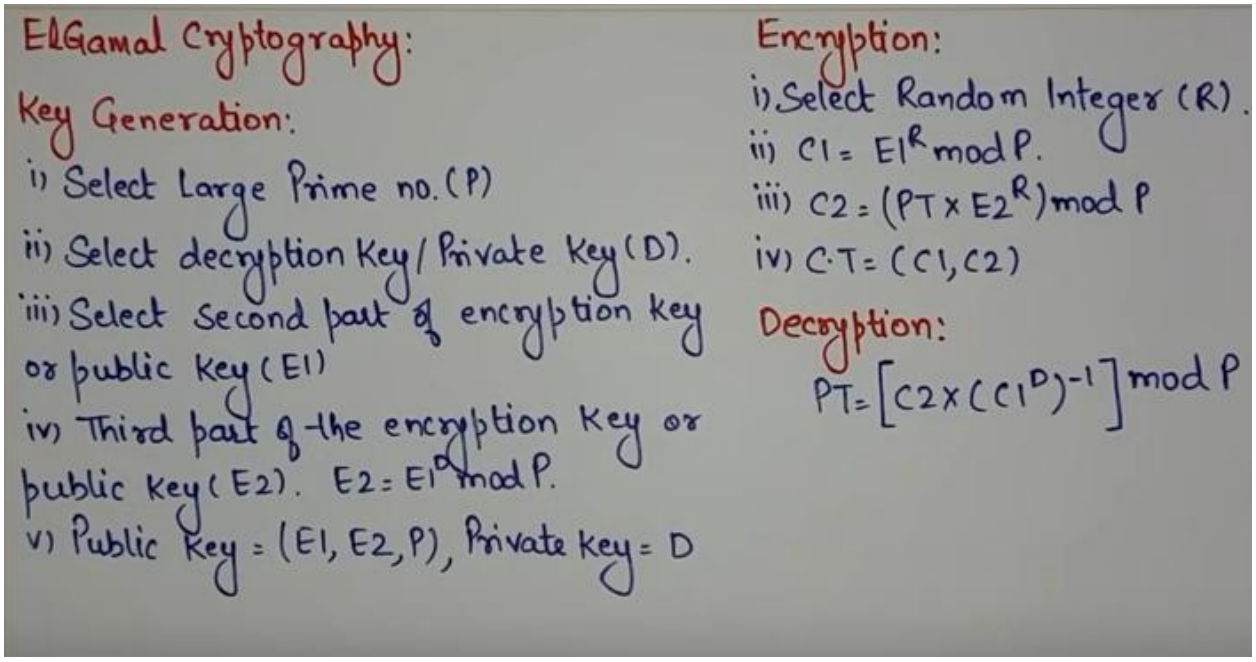
$$= P_m$$

So receiver gets the same point

Comparision of RSA/DSA (Diffie Hellman Algorithm

Key sizes with equivalent security levels

Minimum size (bits) of public keys		
DSA/DH	RSA	ECC
1024	1024	160
2048	2048	224
3072	3072	256
7680	7680	384
15360	15360	512



Elgamal Cryptography: Asymmetruc Key (Encrp-pub key ,Decrypt-Private key)

`Let P=11;D=3;E1=2

$E2=E1^D \bmod p$

$E2= 2^3 \bmod 11$

E2= 8 mod 11

E2=8

Now public key={E1,E2,P}={2,8,11}

Private Key=3

Encryption:

R=4

$C1=E1^R \bmod p$

$C1=2^4 \bmod 11$

C1=16 mod 11=5

Now Pt=7

C2={Pt $E2^R$ mod p}

C2= $7x8^4$ mod 11

C2=28672 mod 11 =6

C2=6

C.T ={5,6}

Decryption

Pt={C2 $(C1^D)^{-1}$ mod p}

Pt={6x$(5^3)^{-1}$)mod 11

First: $(5^3)^{-1}$)mod 11

$(125)^{-1}$)mod 11

125 *X mod 11=1

125*1 mod 11= 121 +4 mod 11=4

125*2 mod 11= 250 mod 11= 242+8 mod11=8

125*3 mod 11=375 mod 11= 374+1 mod 11=1

Pt={6*3 mod 11}

Pt=18 mod 11=7

Pt=7

So the Decrypt and encrypt Ptvalue is same .

UNIT IV MESSAGE AUTHENTICATION AND INTEGRITY

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC – SHA –Digital signature and authentication protocols – DSS
Entity Authentication: Biometrics, Passwords, Challenge Response protocols- Authentication applications – Kerberos, X.509

MESSAGE AUTHENTICATION

- is a mechanism or service used to **verify the integrity of a message**.
- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)

AUTHENTICATION REQUIREMENT

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

Summary:

Message authentication

✓ A procedure to verify that messages come from the alleged (suspected) source and have not been altered

✓ Message authentication may also verify sequencing and timeliness

Digital signature

✓ *An authentication technique* that also includes measures to counter repudiation by either source or destination

I. AUTHENTICATION FUNCTION

Three classes of functions that may be used to produce an authenticator

1)Message encryption – The **Cipher text** of the entire message serves as its authenticator.

2)Message authentication code(MAC) - A public function of the **message and a secret key** that produces a **fixed-length value** that serves as the authenticator.

3) Hashfunction -A public function that maps a **message** of any length into a **fixed-length hash value**, which serves as the authenticator.

1. **MessageEncryption**

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

Symmetric Encryption

- Conventional encryption can serve as authenticator
- Conventional encryption provides *authentication* as well as *confidentiality*
- Message *M* transmitted from source A to destination B is encrypted using a secret key *K* shared by A and B. If no other party knows the key, then confidentiality is provided:
- No other party can recover the plaintext of the message.

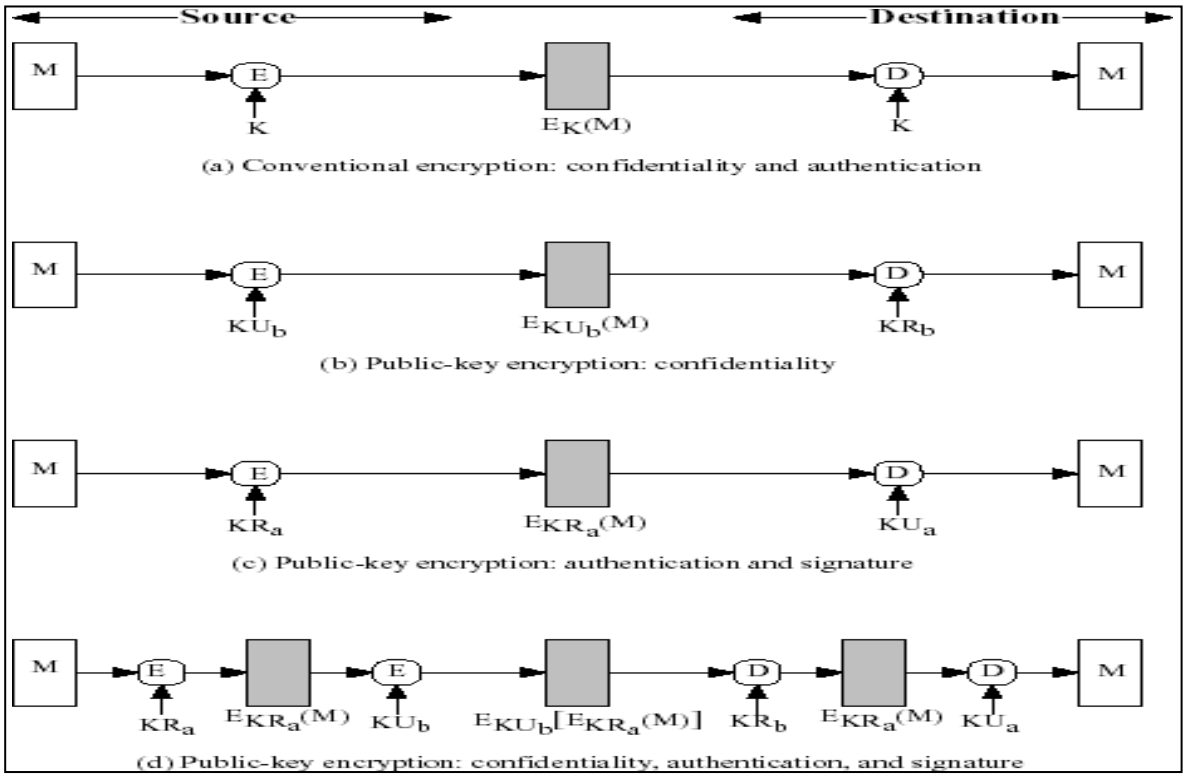


Figure: Basic Uses of Message Encryption

- Given a decryption function D and a secret key *K*, the destination will accept *any* input *X* and produce output $Y = D(K,X)$.
- If *X* is the cipher text of a legitimate message *M* produced by the corresponding encryption function, then *Y* is some plaintext message *M*. Otherwise, *Y* will likely be a meaningless sequence of bits.
- There may need to be some automated means of determining at B whether *Y* is legitimate plaintext and therefore must have come from A.

Public-Key Encryption

- The straightforward use of public-key encryption provides confidentiality but not authentication. The source (A) uses the public key P_{Ub} of the destination (B) to encrypt M. Because only B has the corresponding private key P_{Rb}, only B can decrypt the message.
- To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt.
- To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality.
- The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

(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
$A \rightarrow B: E_{KUb}[E_{KRa}(M)]$ •Provides confidentiality because of KU_b •Provides authentication and signature because of KR_a

Table: Confidentiality and Authentication Implications of Message Encryption

2. <u>Message Authentication Code</u>

- Uses a shared secret key to generate a **fixed-size block of data (known as a cryptographic checksum or MAC)** that is appended to the message.

$$\mathbf{MAC = C_K(M)}$$

Where *M* is a variable-length Input message, *K* is a Shared secret key, C is MAC function and $C_K(M)$ is the fixed-length authenticator.

- The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.
- A MAC function is similar to encryption. One difference is that the **MAC algorithm need not be reversible**, as it must for decryption. In general, the MAC function is a **many-to-one function**.

Assurances

- Message has not been altered
- Message is from alleged sender
- Message sequence is unaltered (requires internal sequencing)

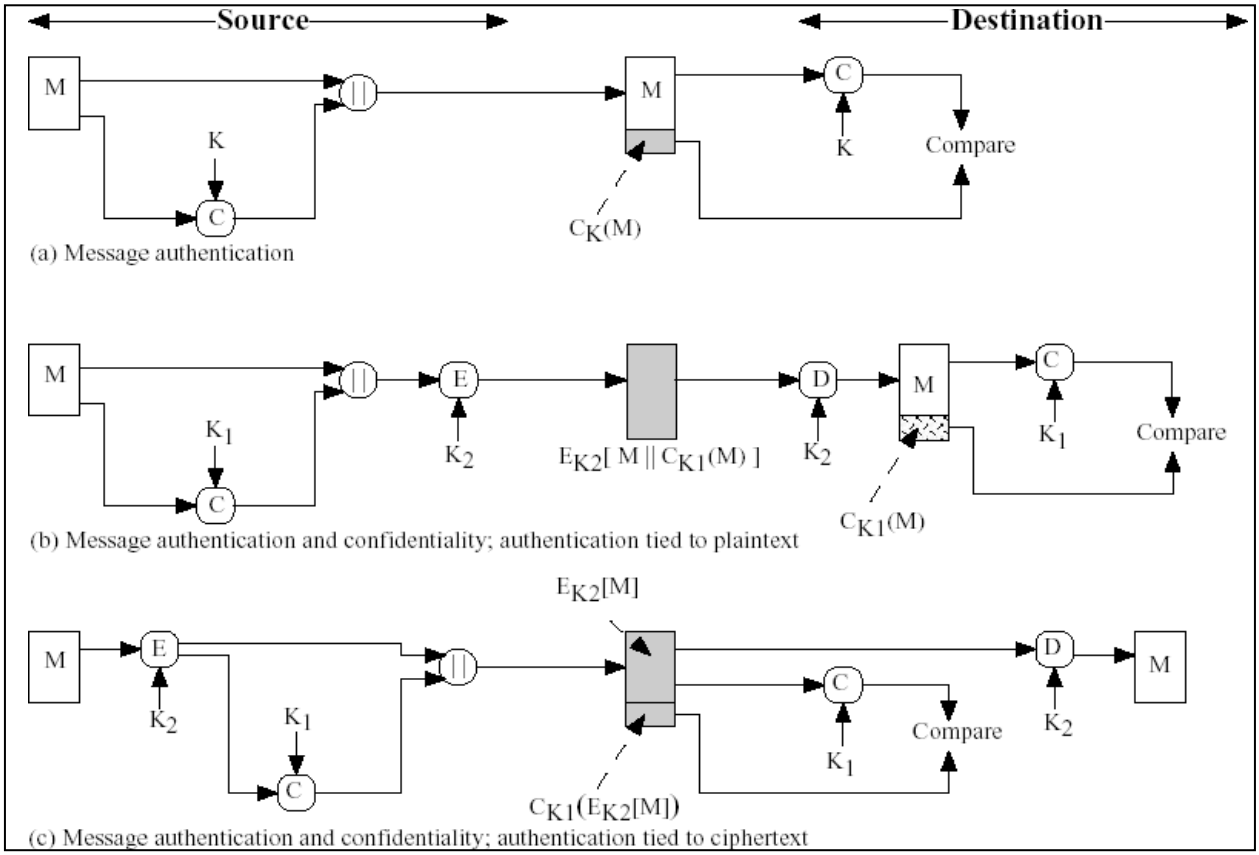


Figure: Basic Uses of MAC

- The process depicted in (Figure a) provides authentication but not confidentiality, because the message as a whole is transmitted in the clear.
- Confidentiality can be provided by performing message encryption either after (Figure b) or before Figure c) the MAC algorithm.
- In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver.
- In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted.
- In the second case, the message is encrypted first.Then the MAC is calculated using the resulting Cipher text and is concatenated to the cipher text to form the transmitted block.
- Note that the MAC does not provide a digital signature because both sender and receiver share the same key.

Table: Basic Uses of Message Authentication Code

(a) $A \rightarrow B: M \| C_K(M)$ •Provides authentication ——Only A and B share K
(b) $A \rightarrow B: E_{K_2}[M \| C_{K_1}(M)]$ •Provides authentication ——Only A and B share K_1 •Provides confidentiality ——Only A and B share K_2
(c) $A \rightarrow B: E_{K_2}[M] \| C_{K_1}(E_{K_2}[M])$ •Provides authentication ——Using K_1 •Provides confidentiality ——Using K_2

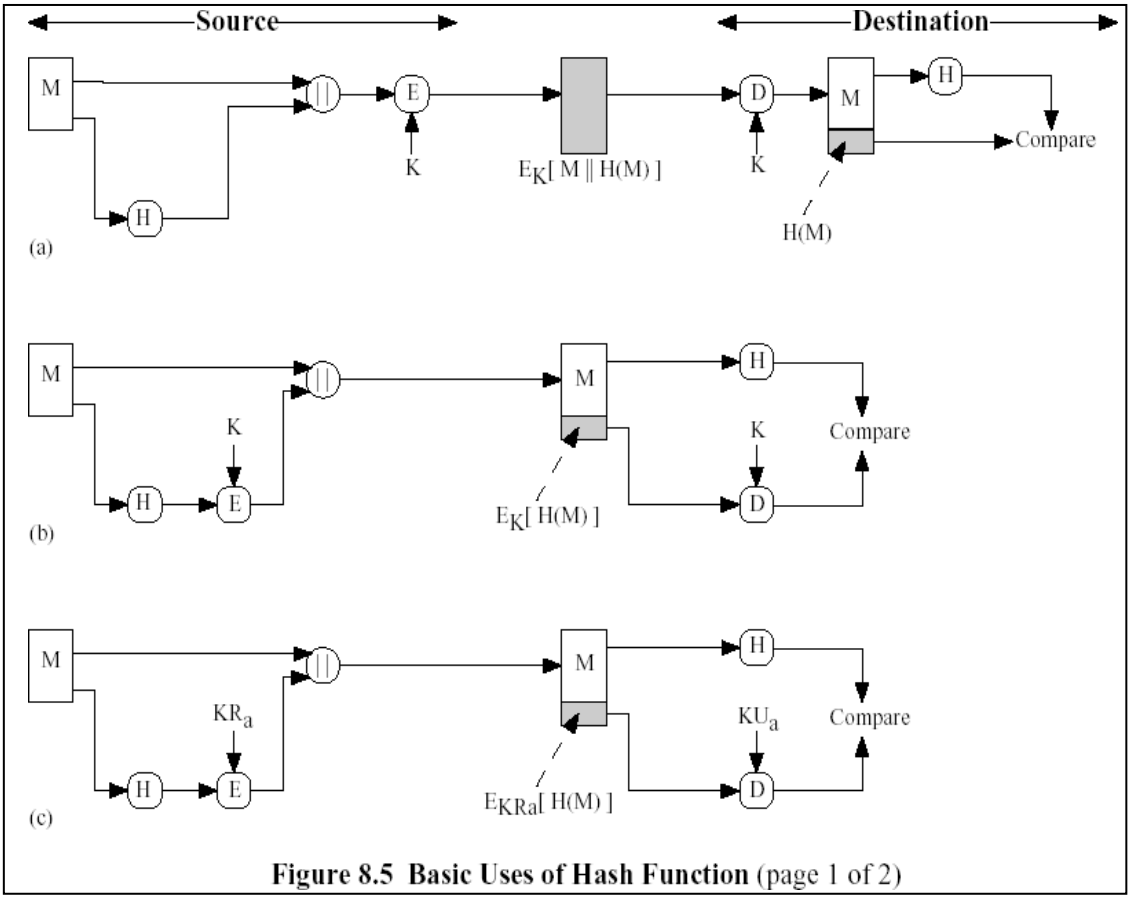


Figure 8.5 Basic Uses of Hash Function (page 1 of 2)

Advantage

- It is cheaper and more reliable to have only one destination responsible for monitoring authenticity.
- Authentication is carried out on a selective basis, messages being chosen at random for checking
- Authentication of a computer program in plaintext is an attractive service.
- Separation of authentication and confidentiality functions affords architectural flexibility
- Separation of authentication check from message use.

3. Hash Function

- **A variation on the message authentication code is the one-way hashfunction**
- **Converts a variable size message M into fixed size hash code H(M) (Sometimes called a message digest)**
- Hash code does not use a key but is a function only of the input message.
- **The hash code (h) is also referred to as a message digest or hash value.**
- The hash code is a function of all the bits of the message and provides an error-detection capability. A change to any bit or bits in the message results in a change to the hash code.

Hash code can be used to provide message authentication, as follows

1. The message plus concatenated hash code is encrypted using symmetric encryption. **E(M ||H)**
2. Only the hash code is encrypted, using symmetric encryption. **M ||E(H)**
3. Only the hash code is encrypted, using public-key encryption and using the sender's private key. M || signed H
4. If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. **E(M || signed H)** gives confidentiality
5. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value *S*. **M || H(M || S)**
6. Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code. E(M || H(M || S))

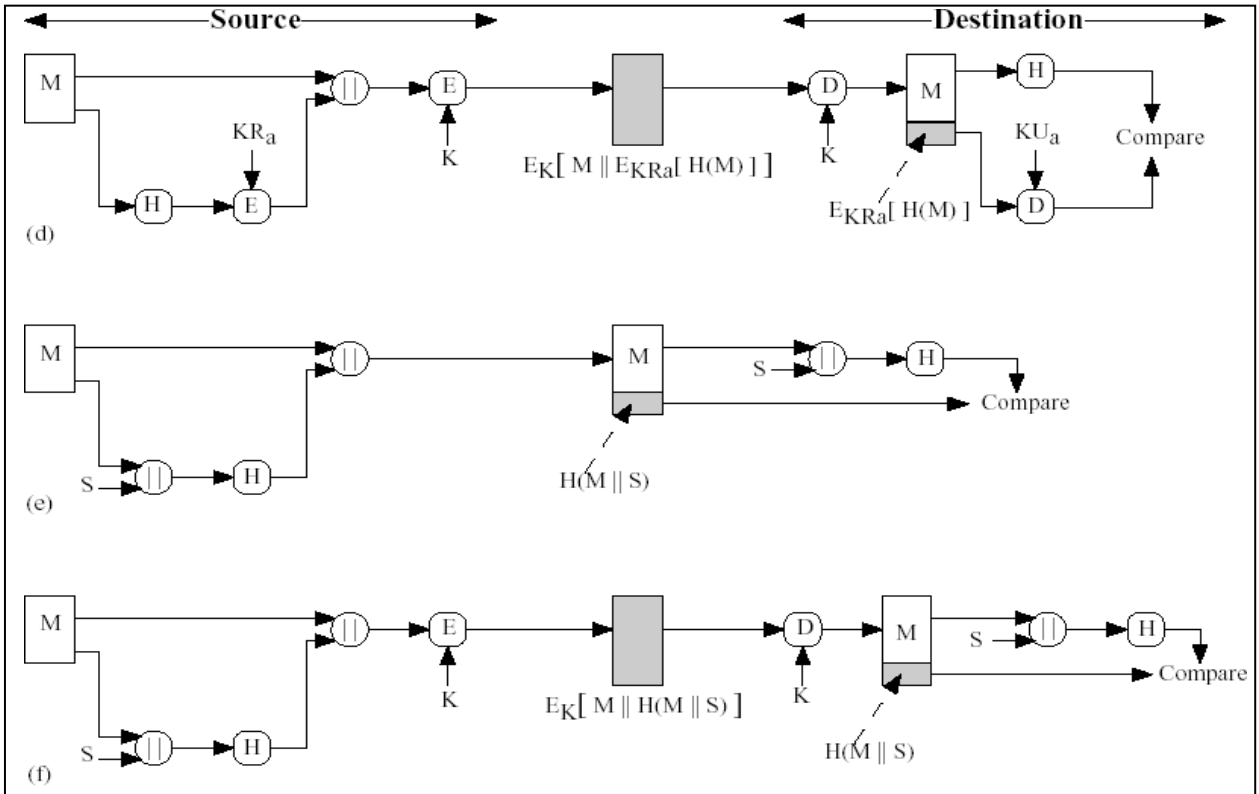


Figure 8.5 Basic Uses of Hash Function (page 2 of 2)

When confidentiality is not required, methods (b) and (c) have an advantage over those that encrypt the entire message in that less computation is required.

Table: Basic Uses of Hash Function H

(a) $A \rightarrow B: E_K[M \| H(M)]$ •Provides confidentiality —Only A and B share K •Provides authentication —H(M) is cryptographically protected	**(d)** $A \rightarrow B: E_K[M \| E_{KRa}[H(M)]]$ •Provides authentication and digital signature •Provides confidentiality —Only A and B share K
(b) $A \rightarrow B: M \| E_K[H(M)]$ •Provides authentication —H(M) is cryptographically protected	**(e)** $A \rightarrow B: M \| H(M \| S)$ •Provides authentication —Only A and B share S
(c) $A \rightarrow B: M \| E_{KRa}[H(M)]$ •Provides authentication and digital signature —H(M) is cryptographically protected —Only A could create $E_{KRa}[H(M)]$	**(f)** $A \rightarrow B: E_K[M \| H(M) \| S]$ •Provides authentication —Only A and B share S •Provides confidentiality —Only A and B share K

Reasons for using Hash

- Encryption software is relatively slow.
- Encryption hardware costs are not negligible.
- Encryption hardware is optimized toward large datasizes.
- Encryption algorithms may be covered by patents

MESSAGE AUTHENTICATION CODES

- A MAC, also known as a cryptographic checksum, is generated by a function C of the form
  T= MAC(K,M)
  Where *M* is a variable-length message, *K* is a secret key shared only by sender and receiver, and MAC(K,*M*) is the fixed-length authenticator, sometimes called a tag.
- The tag is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.

Requirements for MACs:

- Security depends on the **bit length of the key**
- The opponent must resort to a **brute-force attack** using all possible keys.
- On average, such an attack will require $\mathbf{2^{(k-1)}}$ **attempts for a k-bit key**.
- MAC function is a many-to-one function, due to the many-to-one nature of the function.
- Using brute- force methods, how would an opponent attempt to discover a key?
  A number of keys will produce the correct MAC and the opponent has no way of knowing which the correct key is. On average, a total of $2^k/2^n$ = $\mathbf{2^{(k-n)}}$ **keys** will produce a match. Thus, the opponent must iterate the attack:
  - **Round 1**
    - Given: M_1, T_1 = MAC(K, M_1)
    - Compute T_i = MAC(K_i, M_1) for all 2^k keys Number of matches $\approx 2^{(k-n)}$
  - **Round 2**
    - Given: M_2, T_2 = MAC(K, M_2)
    - Compute T_i = MAC(K_i, M_2) for the $2^{(k-n)}$ keys resulting from Round1 Number of matches $\approx 2^{(k-2xn)}$

- If an 80-bit key is used and the MAC is 32 bits long, then the **first round** will produce about 2^{48} possible keys.
- The **second round** will narrow the possible keys to about 2^{16}possibilities.
- The **third round** should produce only a single key, which must be the one used by the sender.
- Brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length.
- Other attacks that do not require the discovery of the key are possible.

The MAC function should satisfy the following requirements:

1. If an opponent observes M and MAC(K, M), it should be computationally infeasible for the opponent to construct a message M' such that MAC(K, M') = MAC(K,M).
2. MAC(K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that MAC(K, M) = MAC(K, M') is 2^{-n}, where n is the number of bits in the tag.
3. Let M' be equal to some known transformation on M. That is, M' = f(M). For example, f may involve inverting one or more specific bits. In that case,
   $$\Pr[MAC(K, M) = MAC(K, M')] = 2^{-n}.$$

- The first requirement speaks about, an opponent is able to construct a new message to match a given tag, even though the opponent does not know and does not learn the key.
- The second requirement deals with the need to prevent a brute-force attack based on chosen plaintext. That is,if we assume that the opponent does not know K but does have access to the MAC function and can present messages for MAC generation, then the opponent could try various messages until finding one that matches a given tag.

HASH FUNCTIONS

A hash value h is generated by a function H of the form

h = H(M)

- M is a variable-length message, h is a fixed-length hash value, H is a hash function
- The hash value is appended at the source
- The receiver authenticates the message by recomputing the hash value
- Because the hash function itself is not considered to be secret, some means is required to protect the hash value

Requirements for a Hash Function

1. H can be applied to any size datablock
2. H produces fixed-length output
3. H(x) is relatively easy to compute for any given x
4. H is *one-way*, i.e., given h, it is computationally infeasible to find an y s.t. H(y) = h
5. H is *weakly collision resistant*: given x, it is computationally infeasible to find any y! ≠ x s.t. H(y) = H(x)
6. H is *strongly collision resistant*: it is computationally infeasible to find any pair (x,y) s.t. H(x) =H(y)
   - One-way property is essential for authentication
   - Weak collision resistance is necessary to prevent forgery
   - Strong collision resistance is important for resistance to birthday attack

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

Simple Hash Functions

- **Operation of hash functions:**
  - The input is viewed as a sequence of n-bit blocks
  - The input is processed one block at a time in an iterative fashion to produce an n-bit hash function

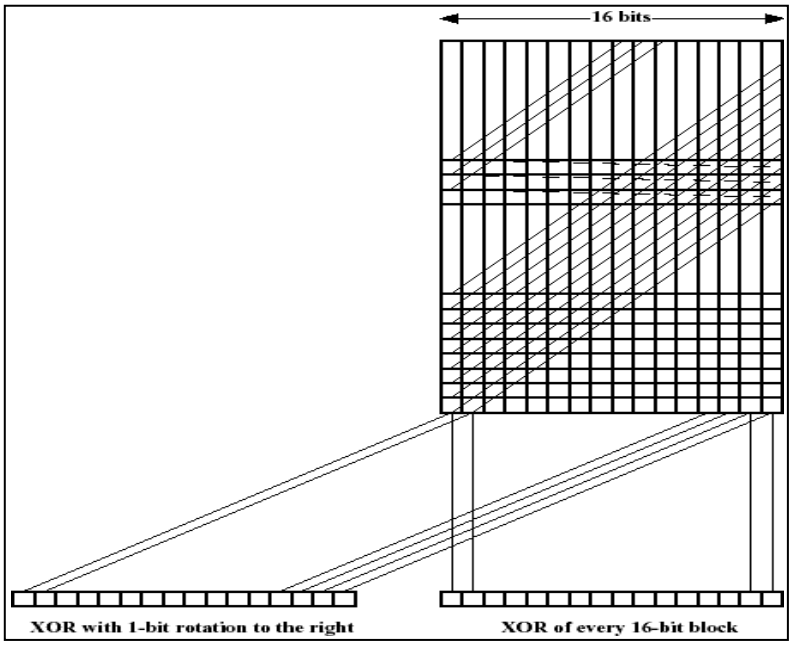
- **Simplest hash function:**
  1) Bitwise XOR of every block
     - $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$

       Where C_i = i-th bit of the hash code, $1 \leq i \leq n$

       m = number of n-bit blocks in the input

       b_{ij} = i-th bit in j-th block

       Known as **longitudinal redundancy check**



Improvement over the simple bitwise XOR

- Initially set the n-bit hash value to zero
- Process each successive n-bit block of data as follows
  - Rotate the current hash value to the left by one bit
  - XOR the block into the hash value

This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input. The above Figure illustrates these two types of hash functions for 16-bit hash values.

We can define the scheme as follows:
Given a message consisting of a sequence of 64-bit blocks $X_1, X_2, \ldots, X_N$, define the hash code C as the block-by-block XOR of all blocks and append the hash code as the final block:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Next, encrypt the entire message plus hash code, using CBC mode to produce the encrypted message $Y_1, Y_2, \ldots, Y_{N+1}$.
Cipher text of this message can be manipulated in such a way that it is not detectable by the hash code.
For example,

$X_1 = IV \oplus D(K, Y_1)$

$X_i = Y_{i1} \oplus D(K, Y_i)$

$X_{N+1} = Y_N \oplus D(K, Y_{N+1})$

But X_{N+1} is the hash code:

$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$

$= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \ldots \oplus [Y_{N1} \oplus \ldots \oplus D(K, Y_N)]$

Because the terms in the preceding equation can be XORed in any order, it follows that the hash code would not change if the cipher text blocks were permuted.

Birthday Attacks

1. The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key.
2. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. The opponent prepares an equal number of messages, all of which are variations on the fraudulent message to be substituted for the real one.
3. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.
4. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. **Because the two variations have the same hash code, they will produce the same signature;** the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of 2^3. The generation of many variations that convey the same meaning is not difficult. For example, the opponent could insert a number of "space-space-backspace" character pairs between words throughout the document. Variations couldthen be generated by substituting "space-backspace-space" in selected instances. Alternatively, the opponent could simply reword the message but retain the meaning.

SECURITY OF HASH FUNCTIONS AND MACS

We can group attacks on hash functions and MACs into two categories: **brute-force attacks and cryptanalysis**.

Brute-Force Attacks

The nature of brute-force attacks differs somewhat for hash functions and MACs.

1) Hash Functions

- H is *one-way*, i.e., given h, it is computationally infeasible to find an y s.t. H(y) = h
- H is *weakly collision resistant*: given x, it is computationally infeasible to find any y! ≠ x s.t. H(y) = H(x)
- H is *strongly collision resistant*: it is computationally infeasible to find any pair (x,y) s.t. H(x) =H(y)

H(x) For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

- One-way and weak collision require 2^neffort
- Strong collision requires $2^{n/2}$effort
- If strong collision resistance is required (and this is desirable for a general-purpose secure hash code), $2^{n/2}$ determines the strength of hash code against brute-force attack
- Currently, two most popular hash codes, SHA-1 and RIPEMD-160, provide a 160-bit hash code length

2) Message Authentication Codes

A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.

Given a fixed message x with n-bit hash code h = H(x), a brute-force method of finding a collision is to pick a random bit string y and check if H(y) = H(x). The attacker can do this repeatedly off line.

Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the key and the MAC.

To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

Computation resistance: Given one or more text-MAC pairs $[x_i, C(K, x_i)]$, it is computationally infeasible to compute any text-MAC pair $[x, C(K, x)]$ for any new input $x \neq x_i$.

In other words, the attacker would like to come up with the valid MAC code for a given message x. There are two lines of attack possible: Attack the key space and attack the MAC value.

Cryptanalysis

Cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search. That is, an ideal hash or MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

Hash Functions

The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs and produces an n-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, b > n; hence the term compression.

The hash function can be summarized as follows:

CV_o = IV = initial n-bit value

CV_i = $f(CV_{i1}, Y_{i1})$ $1 \leq i \leq L$

$H(M)$ = CV_L

where the input to the hash function is a message M consisting of the blocks $Y_o, Y_1,..., Y_{L1.}$

SECURE HASH ALGORITHM (SHA)

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- based on design of MD4 with key differences
- produces 160-bit hash values

REVISED SECURE HASH FUNCTION:

- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1

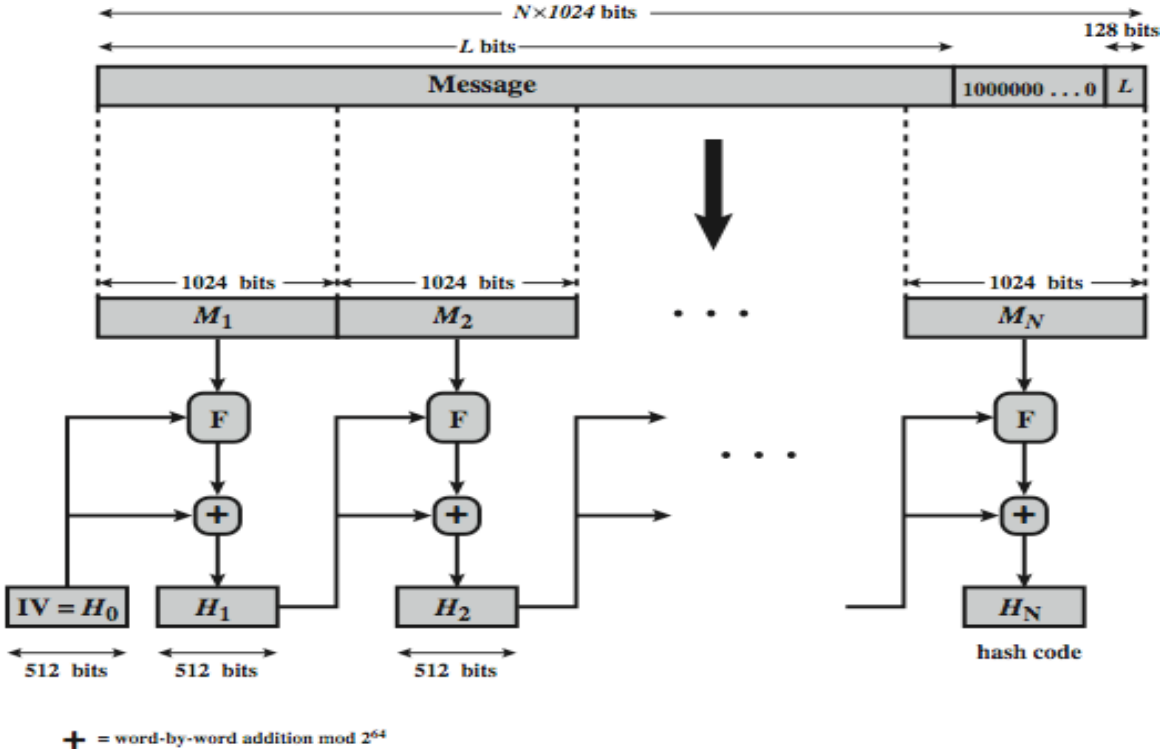
PARAMETERS FOR VARIOUS VERSION OF SHA:

Parameter	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size(in bits)	160	256	384	512
Message size(in bits)	$<2^{64}$	2^{64}	2^{128}	2^{128}
Block size (in bits)	512	512	1024	1024
Word size (in bits)	32	32	64	64
Steps in algorithm	80	64	80	80

SHA-512 logic:

- The input is processed 1024 bits block.
- The algorithm takes as input a message with a maximum length of less than 2^128 bits.
- Produce output is 512 bits message digest.

Algorithm processing Steps: The processing consists of the following steps:
Step 1: Append padding bits
Step 2: Append length
Step 3: Initialize hash buffer
Step 4: Process the message in 1024-bit (128-word) blocks, which forms the heart of the algorithm
Step 5: Output the final state value as the resulting hash



Step-1: Appending Padding Bits. The original message is "padded" (extended) so that its length (in bits) consists of a single 1-bit followed by the necessary number of 0-bits, so that its length is congruent to 896 modulo 1024 (128 bits short of a multiple of 1024)

Step-2: Append length**:** a block of 64 bits is appended to the message. This block is treated as unsigned 64 bit integers (most significant byte first) and contains the length of the original message.

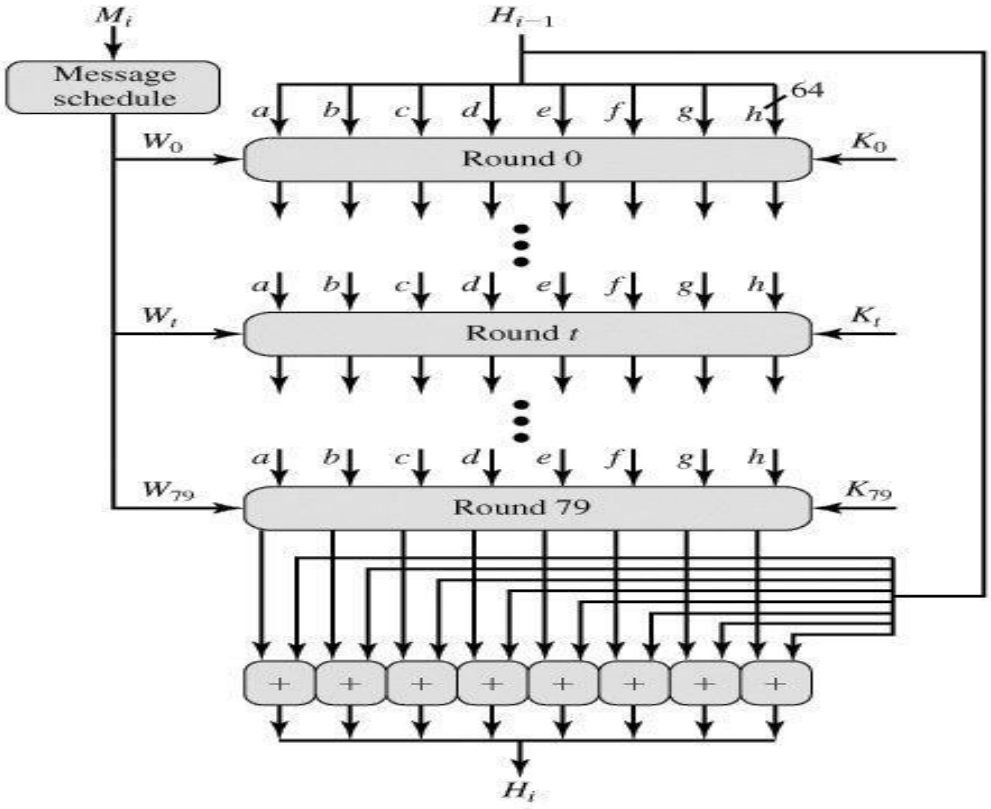
Step-3: Initialize hash buffer: 160 bit buffer is used to hold intermediate and final results of the hash function. This buffer can be represented as eight 64 bit registers (a, b, c, d, e, f, g, h). The registers are initialized to the following 64 bit integers

Word a:6A09E66713BCC908 Word e:BB67AE8584CAA73B
Word b:3C6EF372FE94F82B Word f:A54FF53A5F1D36F1
Word c:510E527FADE682D1 Word g:9B05688C2B3E6C1F
Word d:1F83D9ABFB41BD6B Word h:5BE0CD19137E2179

which are the beginnings, in hexadecimal, of the fractional parts of the square roots of 2, 3, 5, 7, 11, 13, 17, and 19.

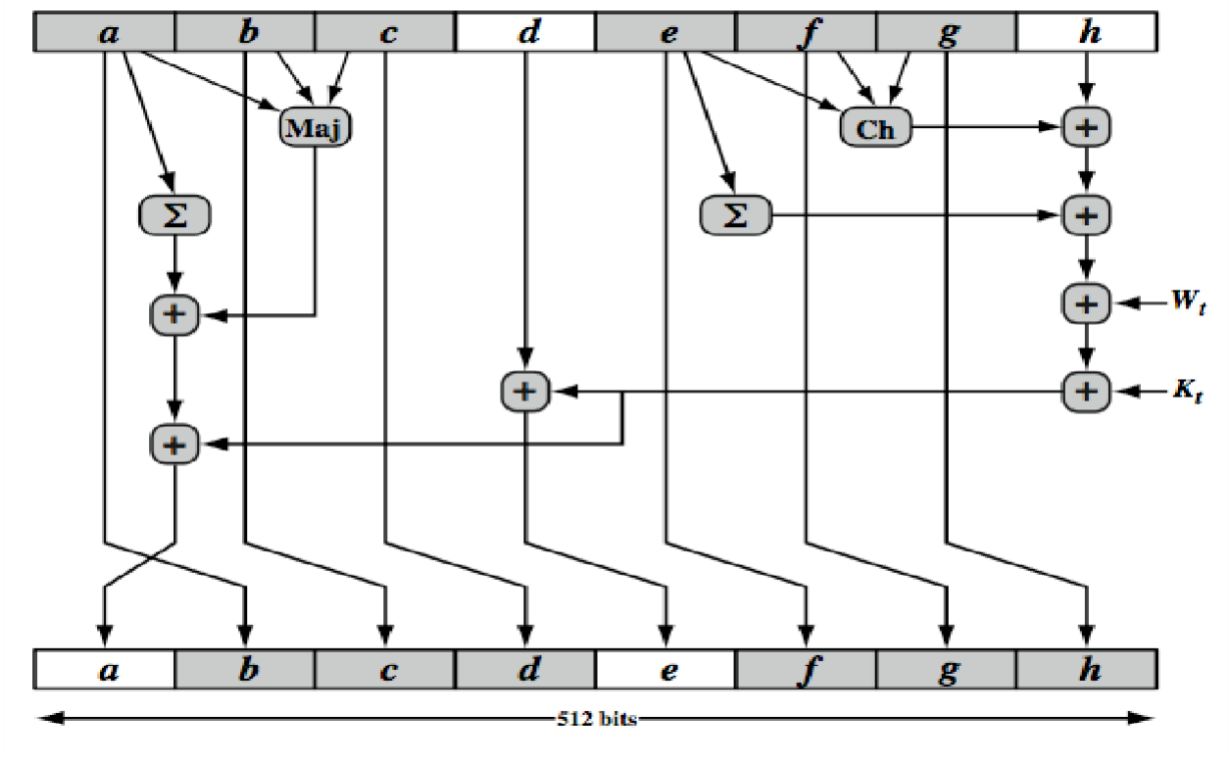
Step 4: Process Message in 512 bits:

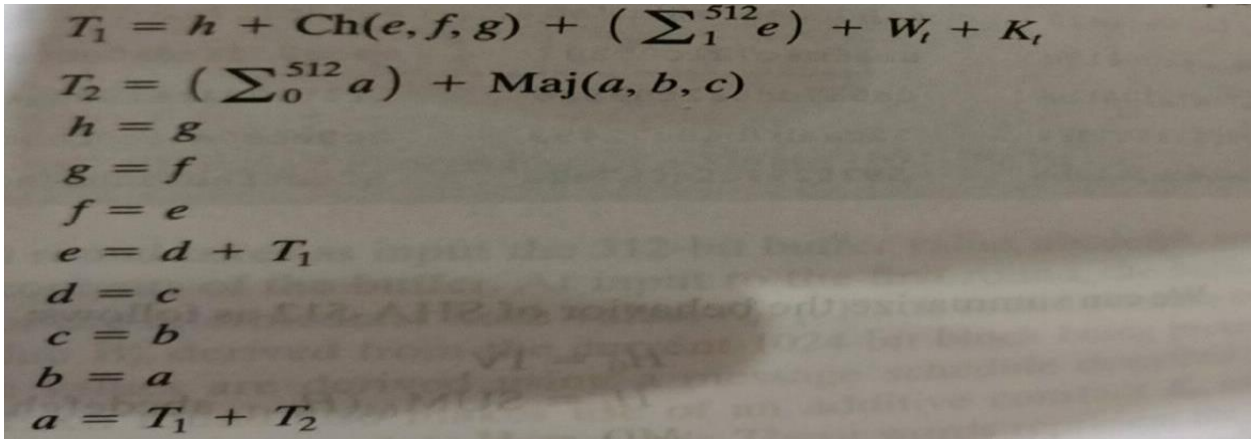
This algorithm consist 4 rounds of 20 steps each. The SHA-512 Compression Function is the heart ofthe algorithm. In this Step 4, it processes the message in 1024-bit (128-word) blocks, using a module that consists of 80 rounds, labeled F in above Figure. Each round takes as input the 512-bit buffer value, and updates the contents of the buffer. Each round t makes use of a 64-bit value Wt derived using a message schedule from the current 1024-bit block being processed. Each round also makes use of an additive constant Kt, based on the fractional parts of the cube roots of the first eighty prime numbers. The output of the eightieth round is added to the input to the first round to produce the final hash value for this message block, which forms the input to the next iteration of this compression function,



SHA-512 Elementary SHA operation for single round (or) SHA-1 Compression Function:

The logic in each of the 80 steps of the processing of one 512 bit block and the structure of each of the 80 rounds is shown Figure.





$$T_1 = h + \mathrm{Ch}(e,f,g) + \left(\sum_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum_0^{512} a\right) + \mathrm{Maj}(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

Each 64-bit word shuffled along one place, and in some cases manipulated using a series of simple logical functions (ANDs, NOTs, ORs, XORs, ROTates), in order to provide the avalanche & completeness properties of the hash function.

The elements are:

Where the elements are:

t=step number; 0<=t<=79

Ch(e,f,g) = (e AND f) XOR (NOT e AND g)

The conditional function: IF e then f else g

Maj(a,b,c) = (a AND b) XOR (a AND c) XOR (b AND c)

The function is true only of the majority (Two or three) of the arguments are true.

∑(a) = ROTR(a,28) XOR ROTR(a,34) XOR ROTR(a,39)

∑(e) = ROTR(e,14) XOR ROTR(e,18) XOR ROTR(e,41)

ROTR(a,n) = Circular right shift(rotation) of the 64 bit argument x by n bits

+ = addition modulo 2^64

Kt = a 64-bit additive constant

Wt = a 64-bit word derived from the current 512-bit input block.

Step-5: Output: After all N 1024 -bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.

The behavior of SHA-512 can be summarized as:

$H_0 = IV$

$H_i = SUM_{64}(H_{i-1}, a\ b\ c\ d\ e\ f\ g\ h_i)$

$MD = H_N$

IV→ initialize value of the buffers a b c d e f g h, defined in step3.

a b c d e f g h_i → the output of the last round of processing of the i^{th} message block

N→ the number of blocks in the message

SUM_{64}→ addition modulo 2^64 performed separately on each word of the pair of inputs.

MD→ final message digest value

Comparison of MD5 and SHA:

	MD5	**SHA-1**
Message Digest Length	128 bits	160 bits
Basic unit of Processing	512 bits	512 bits
Number of Steps	64 (4 rounds of 16)	80(4 rounds of 20)

Maximum Message Size	∞	264-1 bits
Primitive logical functions	4	4
Additive constants used	64	4
Endian format	Little endian	Big endian

DIGITAL SIGNATURE AND AUTHENTICATION PROTOCOLS

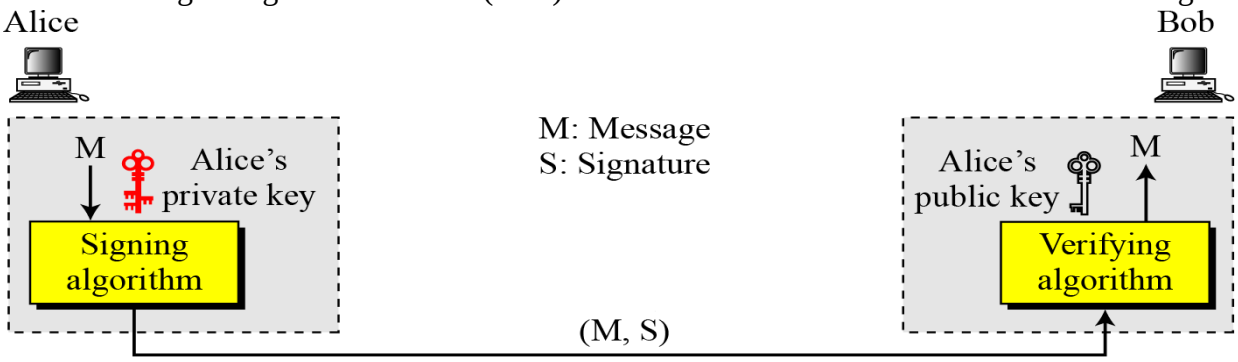
Definition:

A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key. A digital signature or digital signature scheme is a mathematical scheme for demonstration the authenticity of digital message or document.

Means, a digital signature is an authentication mechanism that enables the creator of a message to attach a code that act as a signature.

This signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message.

The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).



Properties of Digital Signature

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

Digital Signature Requirements

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

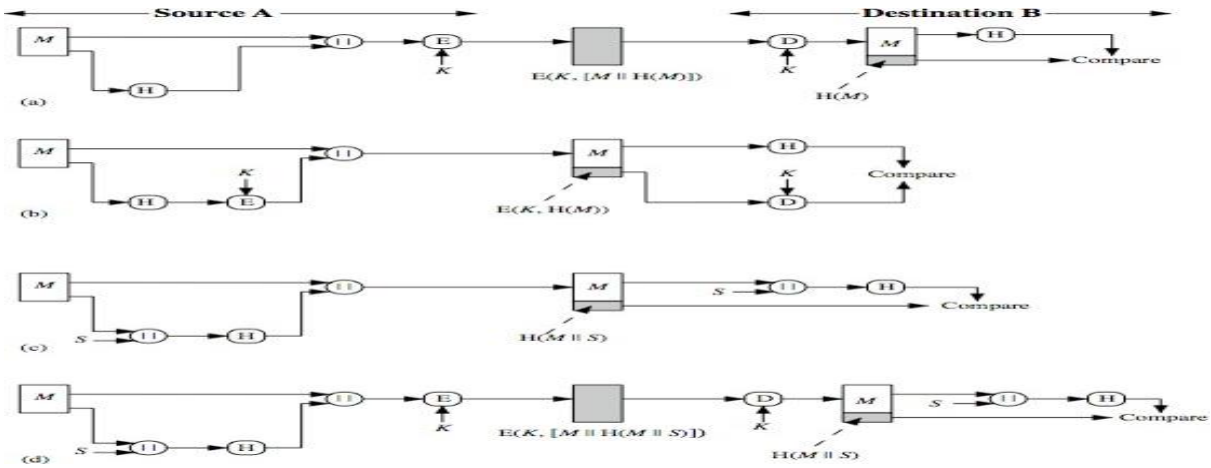
Approaches for Digital Signature

- Direct Digital Signature
- Arbitrated Digital Signature

Direct Digital Signature

The term direct digital signature refers to a digital signature scheme that involves only the communicating parties (source, destination) directly.

- involve only sender & receiver
- assumed receiver has sender's public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key



Arbitrated Digital Signature

In this every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check it origin and content. The message is then dated and sent to Y.

This process is an indication that has been verified to the satisfaction of the arbiter.

By this process, it solves the direct Digital signature problem.

Sender X,

Arbiter A,

Receiver Y,

- X→ construct message M and compute hash value H(M) then X transmitted "M+ Digital Signature" to A.
  Signature consists→ identity "IDx of X +hash value" of all encrypted using K_{XA} (it is common shared key between Sender X and Arbiter A).
- A→ A decrypts the signature & checks the hash value to validate the message. Then transmit it to Y by encryption it with K_{AY} (it is common shared key between Arbiter A and Receiver Y). the message include IDx and M & time Stam.
- Y→ Decrypt it by using K_{AY}

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

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

(c) Public-Key Encryption, Arbiter Does Not See Message

Notations:

X=sender M=message

Y=recipient T=time stamp

A=Arbiter PR_X=X's private key

ID_X=ID of X PU_Y=Y's public key PR_A=A's private key

Digital Signature Standard(DSS)

- US Govt approved signature scheme
- Designed by NIST (National Institute of Standards and Technology) & NSA in early 90's
- Published as Federal Information Processing Standard(FIPS 186) in 1991
- revised in 1993, 1996 & then 2000
- Uses the SHA hash algorithm

- The DSS makes use of the Secure Hash Algorithm (SHA) presents a new digital signature technique, the **Digital Signature Algorithm** (DSA).

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes

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

The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number generated for this particular signature. The signature function also depends on the sender's private key (*PRa*) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G).

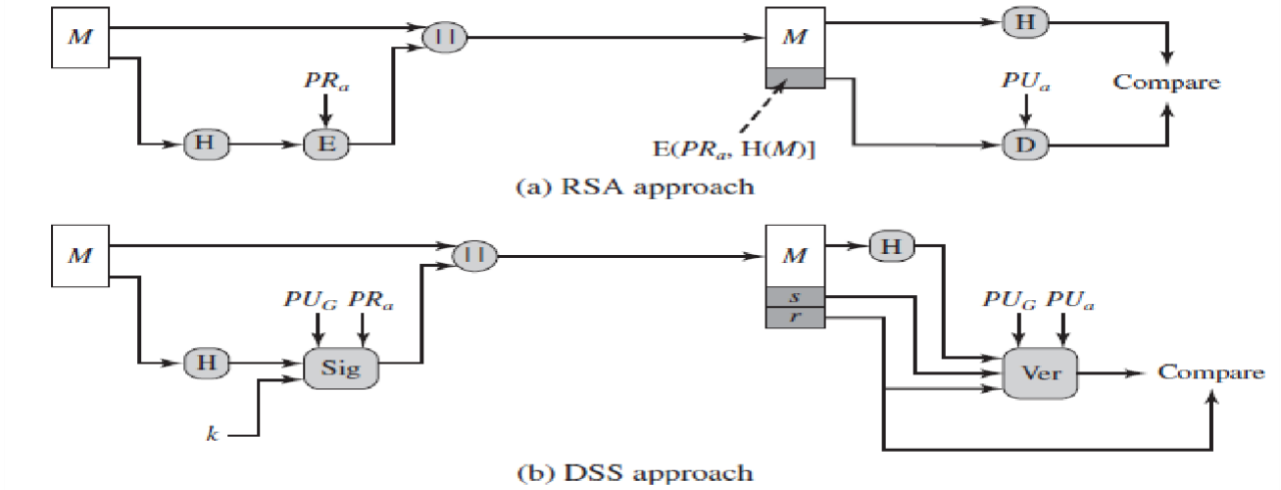


Figure 13.3 Two Approaches to Digital Signatures

Global Public-Key Components

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

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

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

User's Private Key

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

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

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

Signing

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

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

Signature = (r, s)

Verifying

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

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

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

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

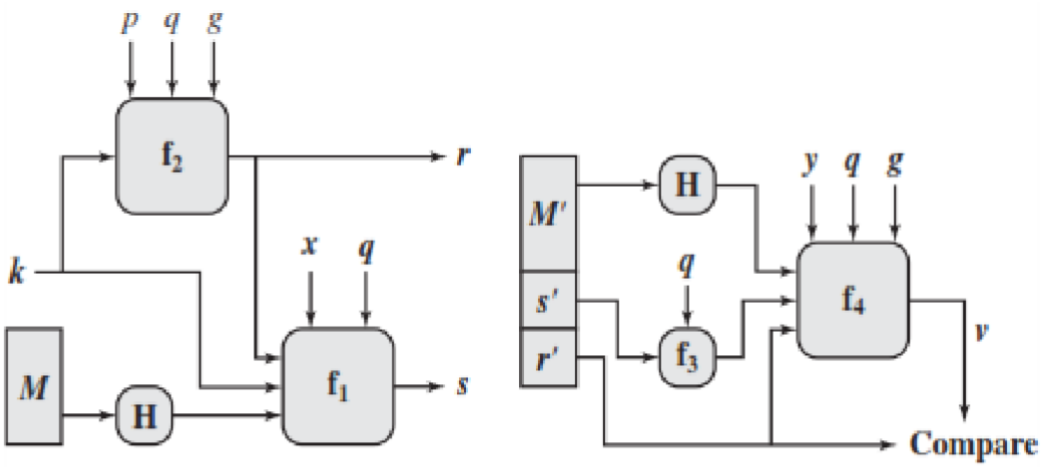
TEST: $v = r'$

M = message to be signed

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

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

Figure 13.4 The Digital Signature Algorithm (DSA)



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Figure 13.5 DSS Signing and Verifying

AUTHENTICATION APPLICATIONS

- It will consider authentication functions
- Its developed to support application-level authentication & digital signatures
    1. Kerberos – a private-key authentication service
    2. X.509 - a public-key directory authentication service

KERBEROS

- Kerberos is an authentication service developed by MIT and is one of the best known and most widely implemented **trusted third party** key distribution systems.
- Provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.
- Two versions of Kerberos are in common use.
    - Version 4
    - Version 5

Kerberos Requirements

- **Secure**: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
- **Reliable**: Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
- **Transparent**: 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.

Kerberos Version 4

- A basic third-party authentication scheme
- Have an Authentication Server (AS)
    - Knows the passwords of all users and stores these in a centralized database.
    - AS shares a unique secret key with each server.
    - These keys have been distributed physically or in some other secure manner
    - users initially negotiate with AS to identify self
    - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- Have a Ticket Granting server (TGS)
    - issues tickets to users who have been authenticated to AS
    - users subsequently request access to other services from TGS on basis of users TGT

Simple Authentication Dialogue

(1) C →AS: $ID_C||P_C||ID_V$

(2) AS →C: Ticket

(3) C →V : $ID_C||$Ticket

$Ticket = E(K_V, [ID_C||AD_C||ID_V])$

Where

C	= client	ID_V	= identifier of V
AS	= authentication server	P_C	= password of user on C
V	=server	AD_C	= network address of C
ID_C	= identifier of user on C	K_V	= secret encryption key shared by AS and V

- The ticket is encrypted to prevent alteration or forgery.
- The server's ID (ID_V) is included in the ticket so that the server can verify that it has decrypted the ticket properly.
- ID_C is included in the ticket to indicate that this ticket has been issued on behalf of C.

The Version 4 Authentication Dialogue

- Obtain ticket granting ticket from AS
  - Once per session
- Obtain service granting ticket from TGT
  - For each distinct service required
- Client/server exchange to obtain service
  - On every service request

Figure provides a simplified overview of the action.

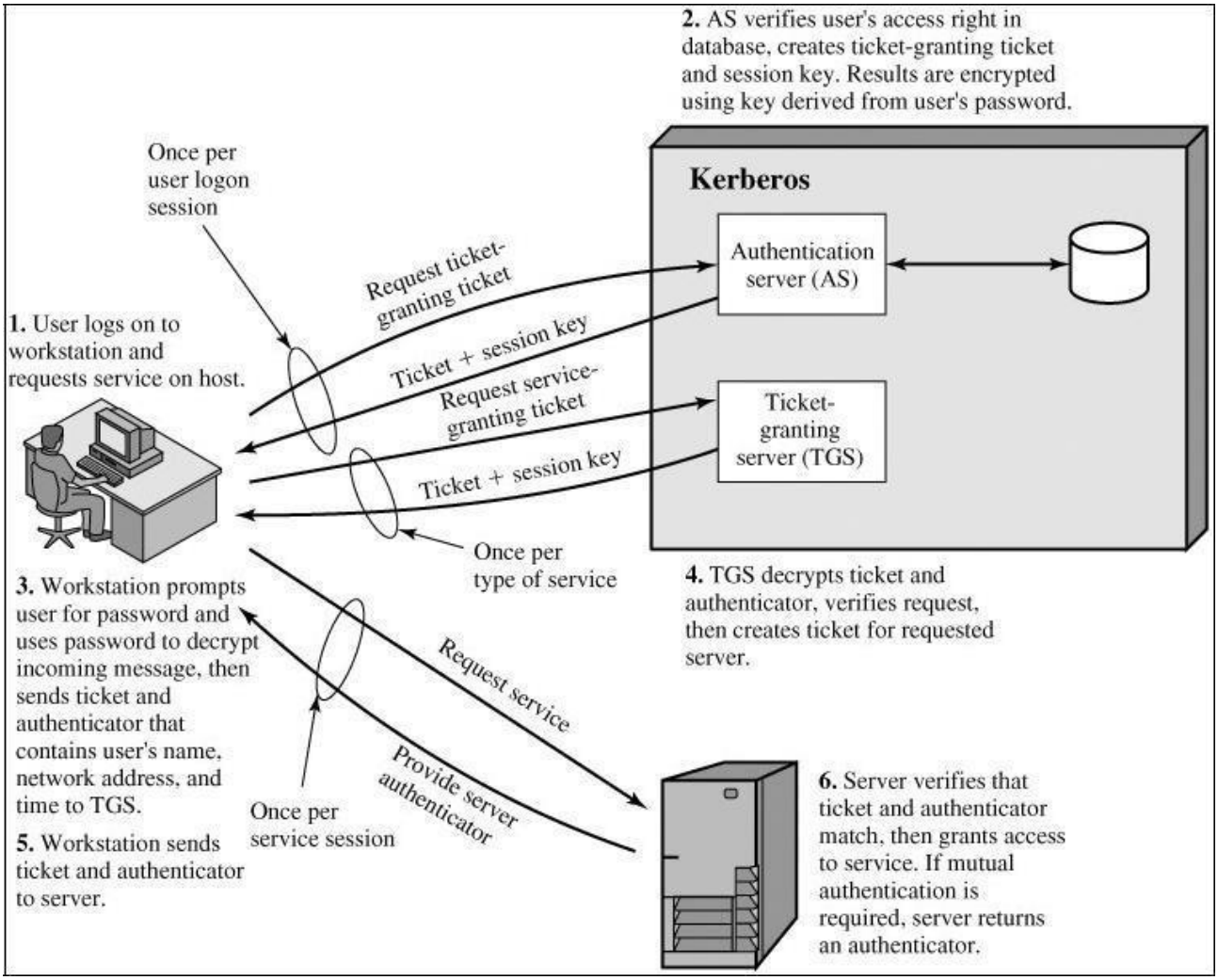


Figure: Overview of Kerberos

- Client sends a message to the AS requesting access to the TGS.
- AS responds with a message, encrypted with a key derived from the user's password (K_C) that contains the ticket.
- Encrypted message also contains a copy of the session key, $K_{C,tgs}$, where the subscripts indicate that this is a session key for C and TGS.
- Session key is inside the message encrypted with K_C, only the user's client can read it.
- Same session key is included in the ticket, which can be read only by the TGS.
- Thus, the session key has been securely delivered to both C and the TGS.
- 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.

Table: Kerberos Version 4 Message Exchanges

(1) C →AS	$ID_C \| ID_{tgs} \| TS_1$
(2) AS →C	$E(K_C, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$
	$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$
Authentication Service Exchange to obtain ticket-granting ticket	
(3) C→TGS	$ID_V \| Ticket_{tgs} \| Authenticator_C$
(4) TGS →C	$E(K_{c,tgs}, [K_{c,v} \| ID_V \| TS_4 \| Ticket_V])$
	$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$ $Ticket_V = E(K_V, [K_{c,v} \| ID_C \| AD_C \| ID_V \| TS_4 \| Lifetime_4])$ $Authenticator_C = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$
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)
	$Ticket_V = E(K_V, [K_{c,v} \| ID_C \| AD_C \| ID_V \| TS_4 \| Lifetime_4])$ $Authenticator_C = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$
Client/Server Authentication Exchange to obtain service	

- The TGS can decrypt the ticket with the key that it shares with the AS. This ticket indicates that user C has been provided with the session key $K_{c,tgs}$. The ticket says, "Anyone who uses $K_{c,tgs}$ must be C."
- The TGS can then check the name and address from the authenticator with that of the ticket and with the network address of the incoming message. If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner.
- The reply from the TGS, in message (4), follows the form of message (2). The message is encrypted with the session key shared by the TGS and C and includes a session key to be shared between C and theserver V, the ID of V, and the timestamp of the ticket. The ticket itself includes the same session key.
- 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)
- The server returns the value of the timestamp from the authenticator, incremented by 1, and encrypted in the session key. C can decrypt this message to recover the incremented timestamp.

Finally, the client and server share a secret key. This key can be used to encrypt future messages between the two or to exchange a new random session key for that purpose.

Kerberos Realms

Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.
3. The Kerberos server in each interoperating realm shares a secret key with the server in the other

realm. The two Kerberos servers are registered with each other.

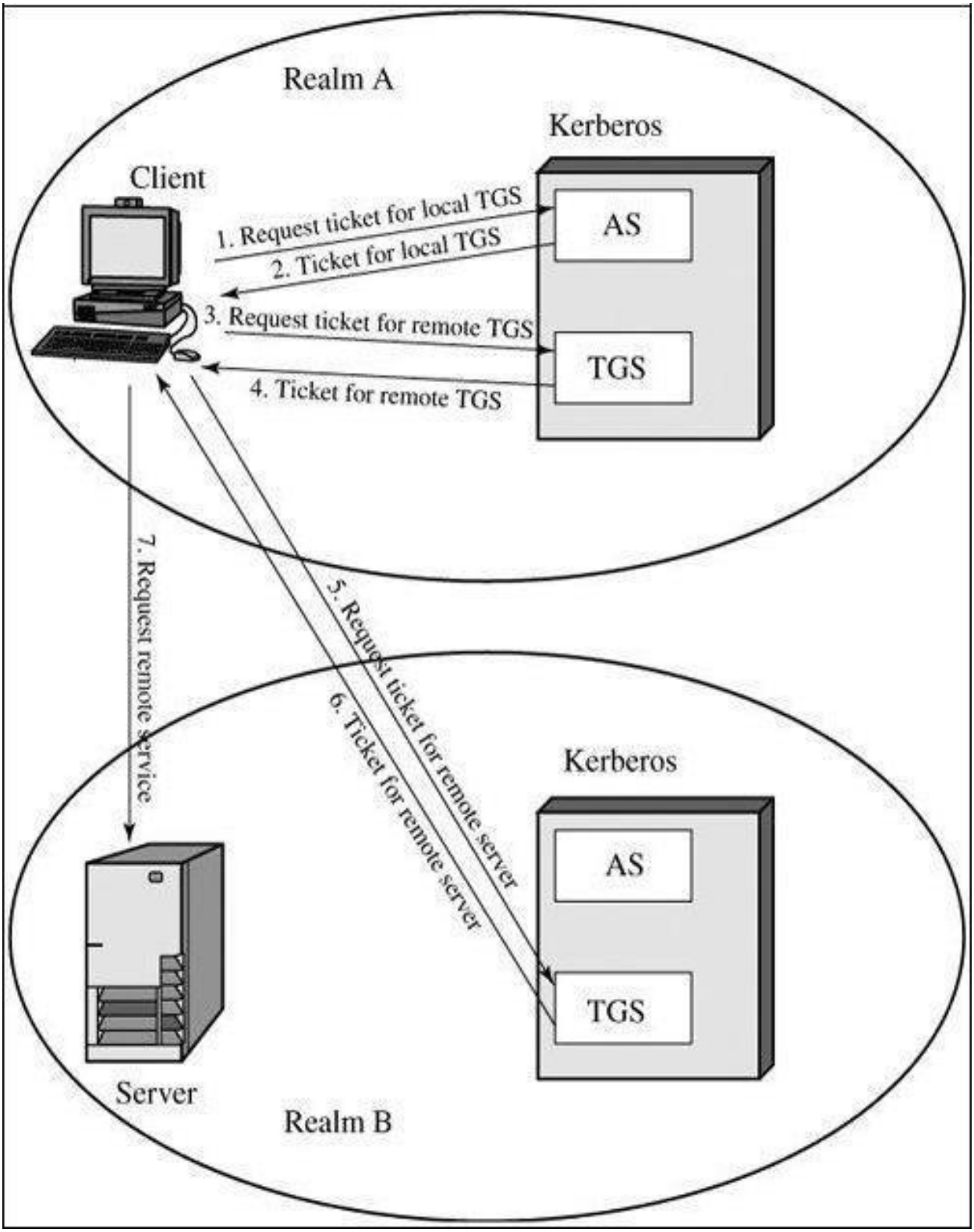


Figure: Request for Service in Another Realm

Such an environment is referred to as a **Kerberos realm**. The concept of *realm* can be explained as follows. A Kerberos realm is a set of managed nodes that share the same Kerberos database.

Kerberos principal, which is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: **a service or user name, an instance name, and a realm name**

A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm). The client can then apply to the remote TGS for a service-granting ticket for the desired server in the realm of the remote TGS.

The ticket presented to the remote server (V_{rem}) indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request.

1) C →AS:	$ID_C \| \| ID_{tgs} \| \| TS_1$
(2) AS →C:	$E(K_C, [K_{c,tgs} \| \| ID_{tgs} \| \| TS_2 \| \| Lifetime_2 \| \| Ticket_{tgs}])$
(3) C →TGS:	$ID_{tgsrem} \| \| Ticket_{tgs} \| \| Authenticator_C$
(4) TGS →C:	$E(K_{c,tgs}, [K_{c,tgsrem} \| \| ID_{tgsrem} \| \| TS_4 \| \| Ticket_{tgsrem}])$
(5) C →TGS_{rem}:	$ID_{vrem} \| \| Ticket_{tgsrem} \| \| Authenticator_C$
(6) TGS_{rem} →C:	$E(K_{C,tgsrem}, [K_{c,vrem} \| \| ID_{vrem} \| \| TS_6 \| \| Ticket_{vrem}])$
(7) C →V_{rem}:	$Ticket_{vrem} \| \| Authenticator_C$

Kerberos Version 5

- Developed in mid 1990's
- Specified as internet standard rfc 1510
- Provides improvements over V4

Differences between Versions 4 and 5

1. **Kerberos Version 4 Environmental shortcomings**
    1. **Encryption system dependence:** Version 4 requires the use of **DES**. In version 5, ciphertext is tagged with an encryption type identifier so that **any encryption technique** may be used.
    2. **Internet protocol dependence:** Version 4 requires the use of **Internet Protocol (IP) addresses**. Version 5 **network addresses are tagged with type and length, allowing any network address type to be used.**
    3. **Message byte ordering:** In version 4, the sender of a message employs a **byte ordering** of its own choosing. In version 5, all message structures are defined using **Abstract Syntax Notation One (ASN.1)**
    4. **Ticket lifetime:** Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is 2^8 x 5 = **1280 minutes**, or a little over 21 hours. In version 5, tickets include an **explicit start time and end time, allowing tickets with arbitrary lifetimes.**
    5. **Authentication forwarding:** Version 4 **does not allow credentials** issued to one client to be forwarded to some other host and used by some other client. Version 5 provides this capability.
    6. **Interrealm authentication:** In version 4, **interoperability among N realms** requires on the order of N^2 **Kerberos-to-Kerberos relationships**, as described earlier. Version 5 supports a method that requires fewer relationships, as described shortly.

2. **Technical Deficiencies**
    1. **Double encryption:** [messages (2) and (4)] that tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. **The second encryption is not necessary and is computationally wasteful.**
    2. **PCBC encryption:** Encryption in version 4 makes use of a nonstandard mode of DES known as **propagating cipher block chaining (PCBC).** Version 5 provides **explicit integrity mechanisms, allowing the standard CBC mode to be used for encryption.**
    3. **Session keys:** Each ticket includes a **session key** that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In version 5, it is possible for a client and server to **negotiate a subsession key, which is to be used only for that one connection.**
    4. **Password attacks:** Both versions **are vulnerable to a password attack**. Version 5 does provide a mechanism known as **preauthentication**, which should make **password attacks more difficult**, but it **does not prevent them.**

SECURITY

The Version 5 Authentication Dialogue

Consider the **authentication service exchange**. Message (1) is a client request for a ticket-granting ticket. As before, it includes the ID of the user and the TGS. The following new elements are added:

- **Realm**: Indicates **realm of user**
- **Options**: Used to request that **certain flags** be set in the returned ticket
- **Times**: Used by the client to request the following time settings in the ticket:
  - **from: the desired start time for the requested ticket**
  - **till: the requested expiration time for the requested ticket**
  - **rtime: requested** renew-till time
- Error! Hyperlink reference not valid.**e**: A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent

Now compare the **ticket-granting service exchange** for versions 4 and 5. We 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).

Summary of Kerberos Version 5 Message Exchanges

(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	$E_{Kc,V}[TS_2 \| \| Subkey \| \| Seq\#]$
	$Ticket_V = E(K_V, [Flags \| \| K_{C,V} \| \| Realm_C \| \| ID_C \| \| AD_C \| \| Times])$ $Authenticator_C = E(K_{C,V}, [ID_C \| \| Realm_C \| \| TS_2 \| \| Subkey \| \| Seq\#])$
(c) Client/Server Authentication Exchange to obtain service	

Client/server authentication exchange, several new features appear in version 5. In message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields as follows:

- **Subkey**: The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket ($K_{C,V}$) is used.
- **Sequence number**: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session.

Ticket Flags

INITIAL	This ticket was issued using the AS protocol
PRE-AUTHENT	Client was authenticated by the KDC before a ticket was issued.
HW-AUTHENT	Use of hardware expected to be possessed solely by the named client.
RENEWABLE	This ticket can be used to obtain a replacement ticket that expires at a later date.
MAY-POSTDATE	Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket.
POSTDATED	Indicates that this ticket has been postdated.
INVALID	This ticket is invalid and must be validated by the KDC before use.
PROXIABLE	Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket.
PROXY	Indicates that this ticket is a proxy.
FORWARDABLE	Tells TGS that a new ticket-granting ticket with a different network address may be issued based on this ticket-granting ticket.
FORWARDED	Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket.

X.509 AUTHENTICATION SERVICE

- ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service
- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users.
- X.509 certificates are widely used and has 3 versions.
- The directory may serve as a repository of public-key certificates of the type.

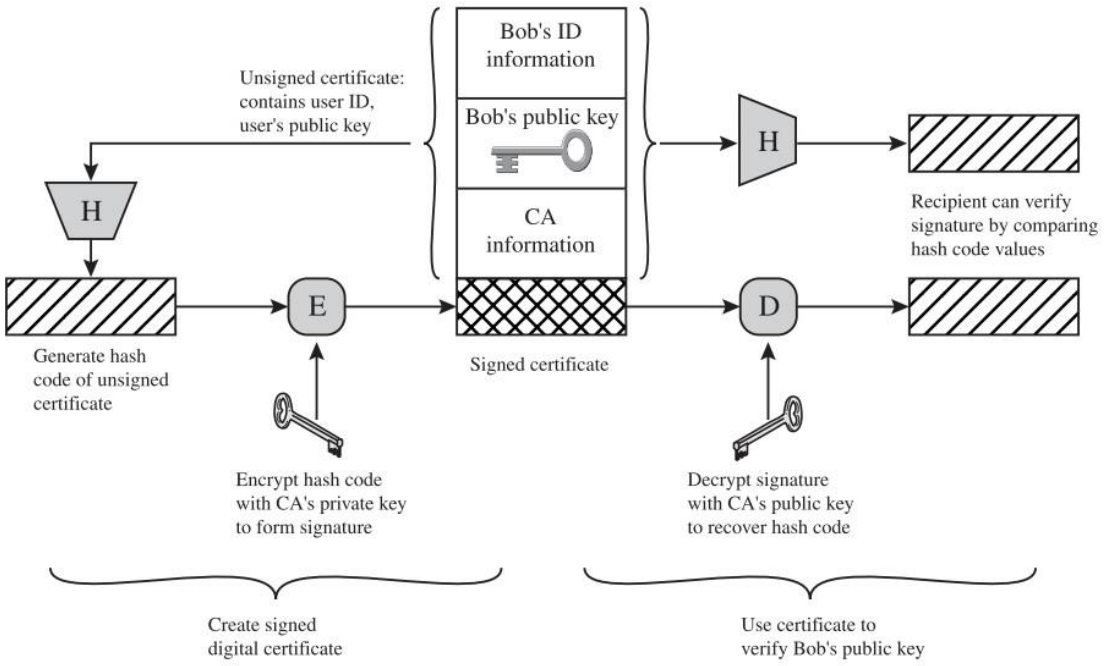


Figure: X.509 Certificate Use

- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority.
- In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

- X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS and SET.
- X.509 is based on the use of public-key cryptography and digital signatures. Algorithms not standardised, but RSA recommended.

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

Certificate includes the following elements:

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

Figure shows the general format of a certificate

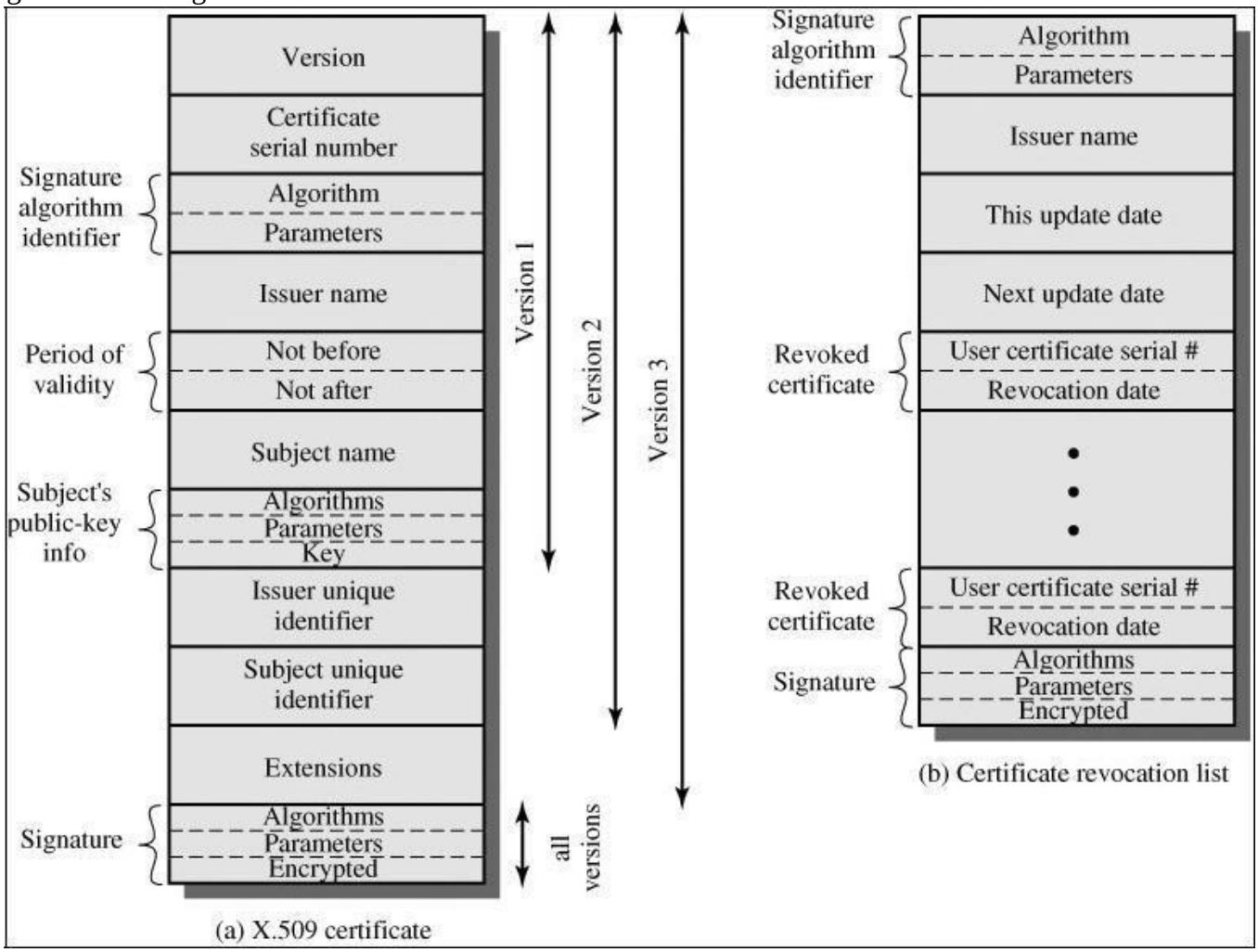


Figure: X.509 Formats

\

The standard uses the following notation to define a certificate:

$CA \ll A \gg = CA \{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$

where

$Y \ll X \gg$ = the certificate of user X issued by certification authority Y
Y {I} = the signing of I by Y. It consists of I with an encrypted hash code appended
V = version of the certificate
SN = serial number of the certificate
AI = identifier of the algorithm used to sign the certificate
CA = name of certificate authority
UCA = optional unique identifier of the CA
A = name of user A
UA = optional unique identifier of the user A
Ap = public key of user A
T^A = period of validity of the certificate

Obtaining a Certificate

User certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can verify the user public key that was certified.
- No party other than the certification authority can modify the certificate without this being detected.

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

CA Hierarchy

- If both users share a common CA then they are assumed to know its public key
- Otherwise CA's must form a hierarchy
- All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public-key certificate.
- X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.
- Use certificates linking members of hierarchy to validate other CA's
- Each CA has certificates for clients (forward) and parent (backward)
    - Each client trusts parents certificates
    - Enable verification of any certificate from one CA by users of all other cas in hierarchy
- The directory entry for each CA includes two types of certificates:
    - **Forward certificates:** Certificates of X generated by other CAs
    - **Reverse certificates:** Certificates generated by X that are the certificates of other CAs

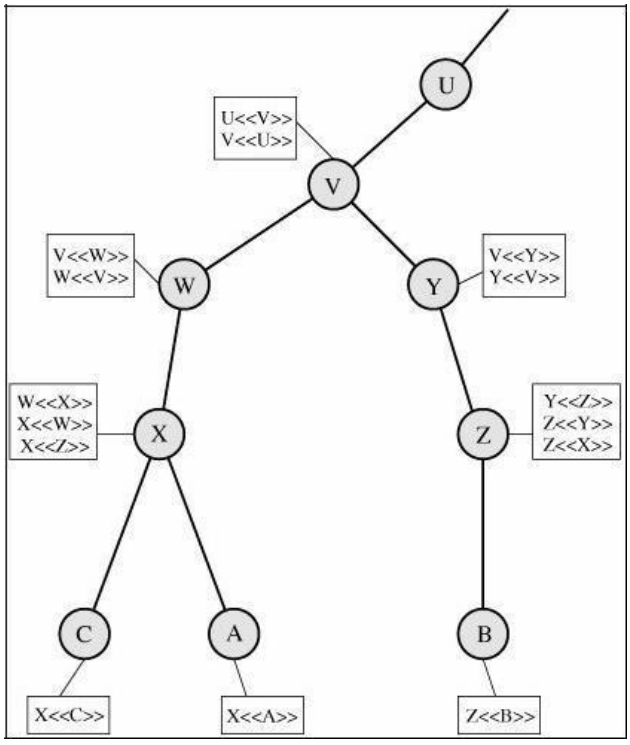


Figure: X.509 Hierarchy: A Example

Track chains of certificates:

A acquires B certificate using chain: X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>
B acquires A certificate using chain: Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

Certificate Revocation

- Certificates have a period of validity
- May need to revoke before expiry, eg:
    - User's private key is compromised
    - User is no longer certified by this CA
    - CA's certificate is compromised

- CA maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs.
- Each certificate revocation list (CRL) posted to the directory is signed by the issuer
- When a user receives a certificate in a message, the user must determine whether the certificate has been revoked.
- The user could check the directory each time a certificate is received.
- To avoid the delays associated with directory searches, it is likely that the user would maintain a local cache of certificates and lists of revoked certificates.

Authentication Procedures

- X.509 also includes three alternative authentication procedures
- 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 following:

1. The identity of A and that the message was generated by A
2. That the message was intended for B
3. The integrity and originality (it has not been sent multiple times) of the message

- Only the identity of the initiating entity is verified in this process, not that of the responding entity.
- Message must include timestamp, nonce, B's identity and is signed by A
- May also be used to convey a session key to B, encrypted with B's public key.

2. Two-Way Authentication

In addition to the three elements just listed, two-way authentication establishes the following elements:

4. The identity of B and that the reply message was generated by B
5. That the message was intended for A
6. The integrity and originality of the reply

- Two-way authentication thus permits both parties in a communication to verify the identity of the other.
- reply includes original nonce from A, also timestamp and nonce from B
- may include additional info for A

3. Three-Way Authentication

- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

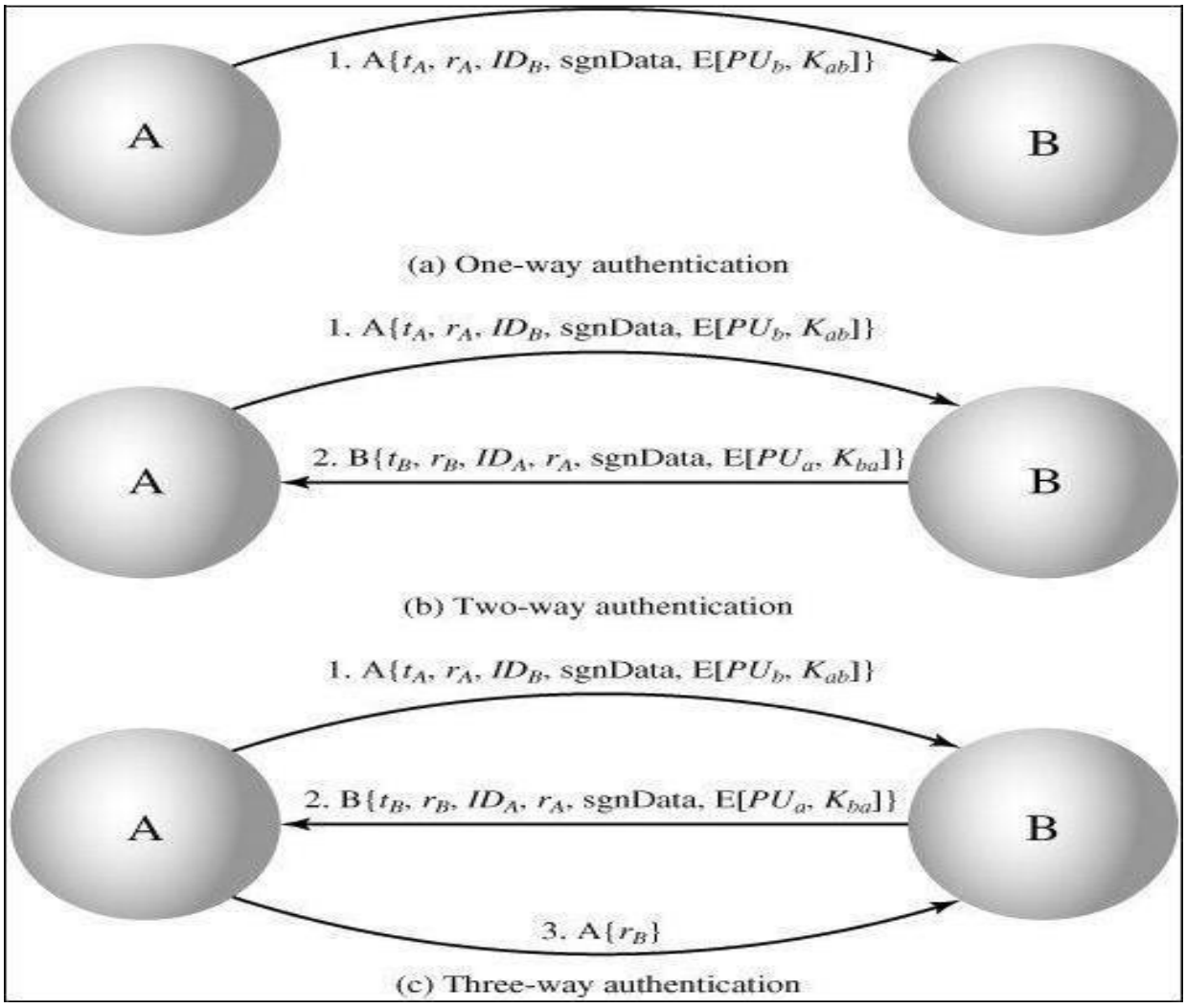


Figure: X.509 Strong Authentication Procedures

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. The following requirements not satisfied by version 2:

1. The Subject field is inadequate to convey the identity of a key owner to a public-key user. X.509 names may be relatively short and lacking in obvious identification details that may be needed by the user.
2. The Subject field is also inadequate for many applications, which typically 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.

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

(1) Key and Policy Information

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy.. For example, a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes the following:

Authority key identifier: Identifies the public key to be used to verify the signature on this certificate or CRL.

Subject key identifier: Identifies the public key being certified.

Key usage: Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used.

Private-key usage period: Indicates the period of use of the private key corresponding to the public key. For example, with digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key.

Certificate policies: Certificates may be used in environments where multiple policies apply.

Policy mappings: Used only in certificates for CAs issued by other CAs.

(2) Certificate Subject and Issuer Attributes

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

The extension fields in this area include the following:

- **Subject alternative name:** Contains one or more alternative names, using any of a variety of forms
- **Subject directory attributes:** Conveys any desired X.500 directory attribute values for the subject of this certificate.

(3) Certification Path Constraints

These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs. The extension fields in this area include the following:

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

UNIT V

SECURITY PRACTICE AND SYSTEM SECURITY

ELECTRONIC MAIL SECURITY – PGP, S/MIME – IP security – Web Security

SYSTEM SECURITY: Intruders – Malicious software – viruses – Firewalls.

SECURITY SERVICES FOR ELECTRONIC MAIL

- ✓ ***Privacy***—the ability to keep anyone but the intended recipient from reading the message.
- ✓ ***Authentication*—**reassurance to the recipient of the identity of the sender.
- ✓ ***Integrity*—**reassurance to the recipient that the message has not been altered since it was transmitted by the sender.
- ✓ ***Non-repudiation***—the ability of the recipient to prove to a third party that the sender really did send the message. This feature is also sometimes called **third party authentication**. The term *non-repudiation* means that the sender cannot later deny sending the message.
- ✓ **Proof of submission**—verification given to the sender that the message was handed to the mail delivery system (the same basic idea as sending certified mail through the U.S. postal service). With certified postal mail you just receive proof that you sent something to a particular address on a particular date, but with electronic mail it is possible to have the mail system verify acceptance of the contents of a particular message, perhaps by signing the message digest of the contents of the message.
- ✓ **Proof of delivery**—verification that the recipient received the message. Postal mail has a similar feature (return receipt requested), but again it only verifies that something was delivered on a particular date to the recipient. With electronic mail it is possible to verify the contents, as we mentioned under proof of submission.
- ✓ **Message flow confidentiality**—an extension of privacy such that Carol not only cannot know the content of the message Alice sent Bob, but cannot even determine whether Alice sent Bob a message. A**nonymity**—the ability to send a message so that the recipient can't find out the identity of the sender.
- ✓ **Containment**—the ability of the network to keep certain security levels of information from leaking out of a particular region.
- ✓ ***Audit***—the ability of the network to record events that might have some security relevance, such as that Alice sent a message to Bob on a particular date. This would be fairly straightforward to implement, but is not mentioned in any of the secure mail standards, so we don't have a section on it.
- ✓ **Accounting**—the ability of the mail system to maintain system usage statistics. In addition to providing clues for system resource management, this information allows the mail system to charge its clients according to their usage. For example, the system might charge by number of messages sent, as long as the system itself authenticates the source of each message to ensure that the proper party is billed. Again, there's not much to say about this, so we don't have a separate section on it.
- ✓ S**elf destruct**—an option allowing a sender to specify that a message should be destroyed after delivery to the recipient. This allows Alice to send a message to Bob that Bob cannot forward or store. The mail system will decrypt and display the message, but then delete it. (*Good morning Mr. Phelps...*). This can be implemented by marking the message as a *self-destruct* message, and having the mail program at the destination cooperate by deleting the message immediately after displaying it.
- ✓ **Message sequence integrity**—reassurance that an entire sequence of messages arrived in the order transmitted, without any loss.

PGP (PRETTY GOOD PRIVACY):

- ➢ **PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.**
- ➢ **Developed by Phil Zimmermann.**

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

1. **It is available free worldwide in versions that run on a variety of platforms.**
2. **The commercial version satisfies users who want a product that comes with vendor support.**

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

4. It has a wide range of applicability.

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

6. PGP is now on an Internet standards track (RFC 3156; MIME Security with OpenPGP). Nevertheless, PGP still has an feeling of an antiestablishment endeavor.

Operational Description:

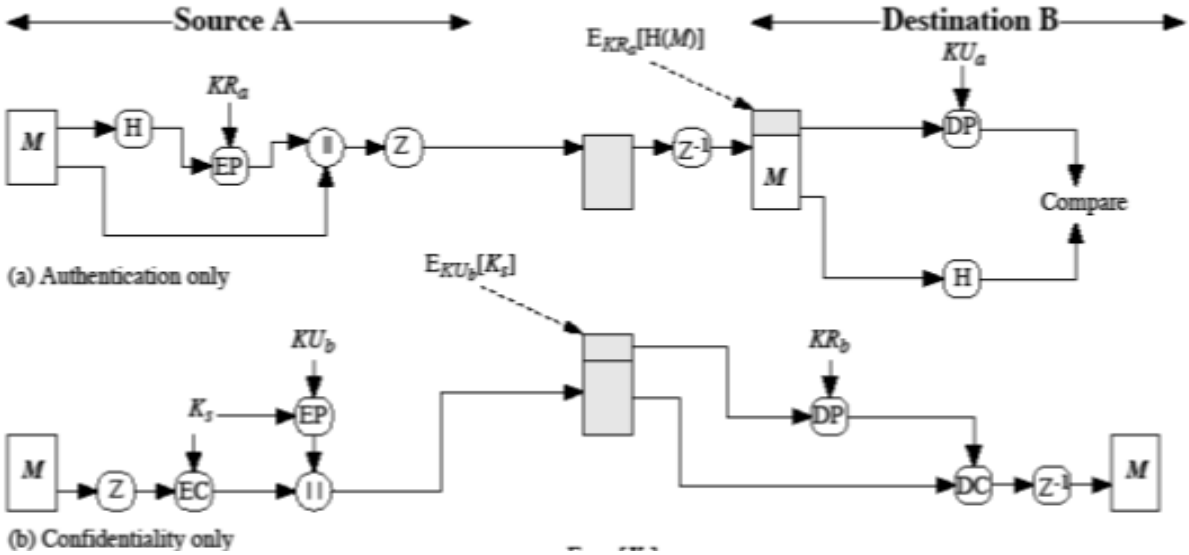
The actual operation of PGP, as opposed to the management of keys, consists of five services: **authentication, confidentiality, compression, e-mail compatibility, and segmentation.**

Authentication:

1. Sender creates message
2. Generates a digital signature for the message
3. Use SHA-1 to generate 160-bit hash of message
4. Signed hash with RSA using sender's private key, and is attached to message
5. Receiver uses RSA with sender's public key to decrypt and recover hash code
6. Receiver verifies received message using hash of it and compares with decrypted hash code

Confidentiality:

1. Sender generates a message and encrypts it.
2. Generates a128-bit random number as session key
3. Encrypts the message using CAST-128 / IDEA / 3DES in CBC mode with session key
4. Session key encrypted using RSA with recipient's public key and attached to the message.
5. Receiver uses RSA with private key to decrypt and recover session key
6. Session key is used to decrypt message



Compression

- ✓ PGP compresses messages to save space for e-mail transmission and storage.
- ✓ By default PGP compresses message after signing but before encrypting
  - o so can store uncompressed message & signature for later verification
  - o Encryption after compression strengthens security (because compression has less redundancy)
- ✓ Uses ZIP compression algorithm
- ✓ **Message encryption** is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

E-mail Compatibility

- ✓ When using PGP will have binary data (8-bit octets) to send (encrypted message, etc)
- ✓ However email was designed only for text
- ✓ Hence PGP must encode raw binary data into printable ASCII characters
- ✓ Uses radix-64 algorithm

- ✓ Maps three octets of binary data are mapped into four ASCII characters and also append a CRC to detect the transmission errors.

Segmentation and Reassembly

- ✓ E-mail facilities often are restricted to a maximum message length. For example, many of the facilities accessible through the Internet impose a maximum length of 50,000 octets.
- ✓ Any message longer than that must be broken up into smaller segments, each of which is mailed separately.
- ✓ To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation is done after all of the other processing, including the radix-64 conversion.

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.

Cryptographic keys and key rings

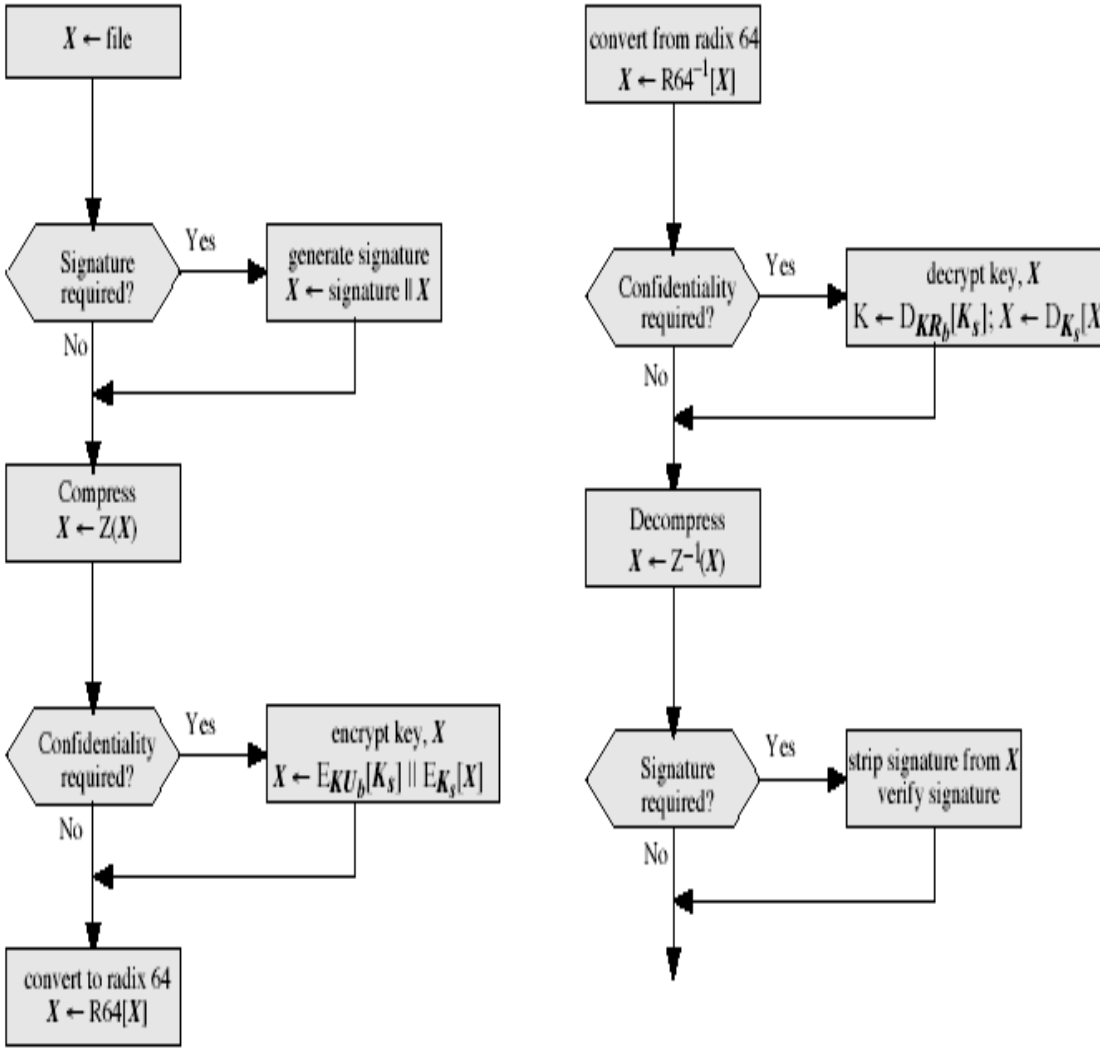
Three separate requirements can be identified with respect to these keys:

- A means of generating unpredictable session keys is needed.
- It must allow a user to have multiple public key/private key pairs.
- Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

Session key generation

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

The input to the random number generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key.



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

Transmission and Reception of PGP messages

The plaintext input to CAST-128 is itself derived from a stream of 128-bit randomized numbers. These numbers are based on the keystroke input from the user.

b. Key identifiers

- Each PGP user has a pair of keyrings:
  - public-key ring contains all the public-keys of other PGP users known to this user, indexed by key ID
  - private-key ring contains the public/private key pair(s) for this user, indexed by key ID & encrypted keyed from a hashed passphrase
- Security of private keys thus depends on the pass-phrase security

PGP MESSAGE FORMAT:

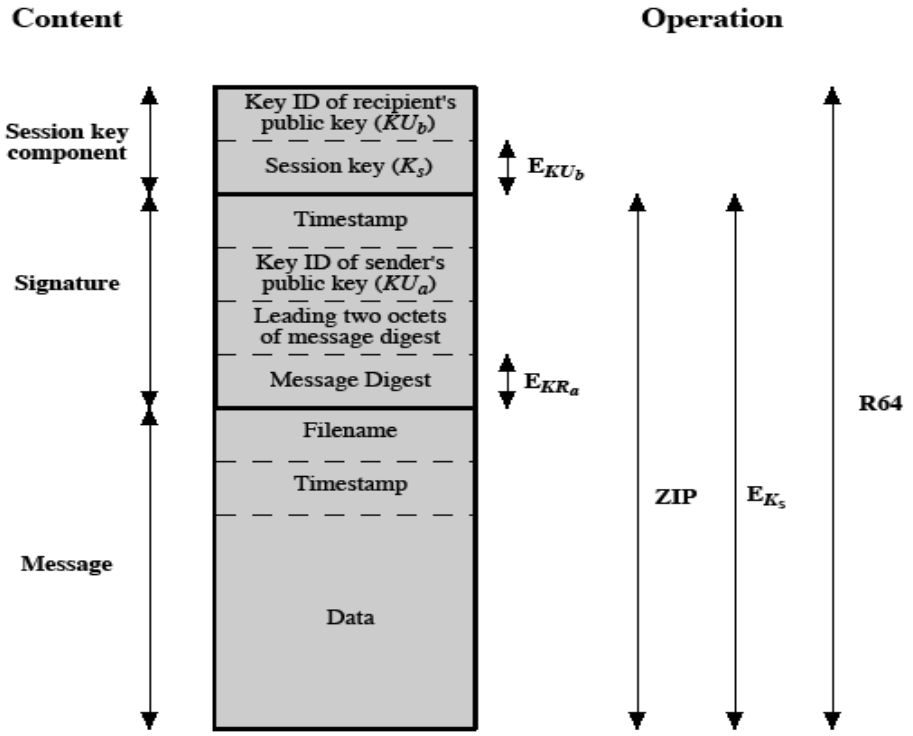
Message consists of three components:

- **Message component** – includes actual data to be transmitted, as well as the filename and a timestamp that specifies the time of creation.
- **Session key component** – includes session key and the identifier of the recipient public key.
- **Signature component** – includes the following
  - **Timestamp** – time at which the signature was made.
  - **Message digest** – hash code.
  - **Two octets of message digest** – to enable the recipient to determine if the correct public key was used to decrypt the message.
  - **Key ID of sender's public key** – identifies the public key

Notation:
EkU_b= encryption with user B's Public key
EKR_a= encryption with user A's private key
EK_s = encryption with session key
ZIP = Zip compression function
R64 = Radix- 64 conversion function

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

The general structures of the private and public key rings are shown below:
Timestamp - the date/time when this entry was made.
Key ID - the least significant bits of the public key.
Public key - public key portion of the pair.
Private Key - private key portion of the pair.
User ID - the owner of the key
Key legitimacy field – indicates the extent to which PGP will trust that this is a valid public key for this user.



General Format of PGP message (From A to B)

Signature trust field – indicates the degree to which this PGP user trusts the signer to certify public key.
Owner trust field - indicates the degree to which this public key is trusted to sign other public key certificates.

PGP message generation

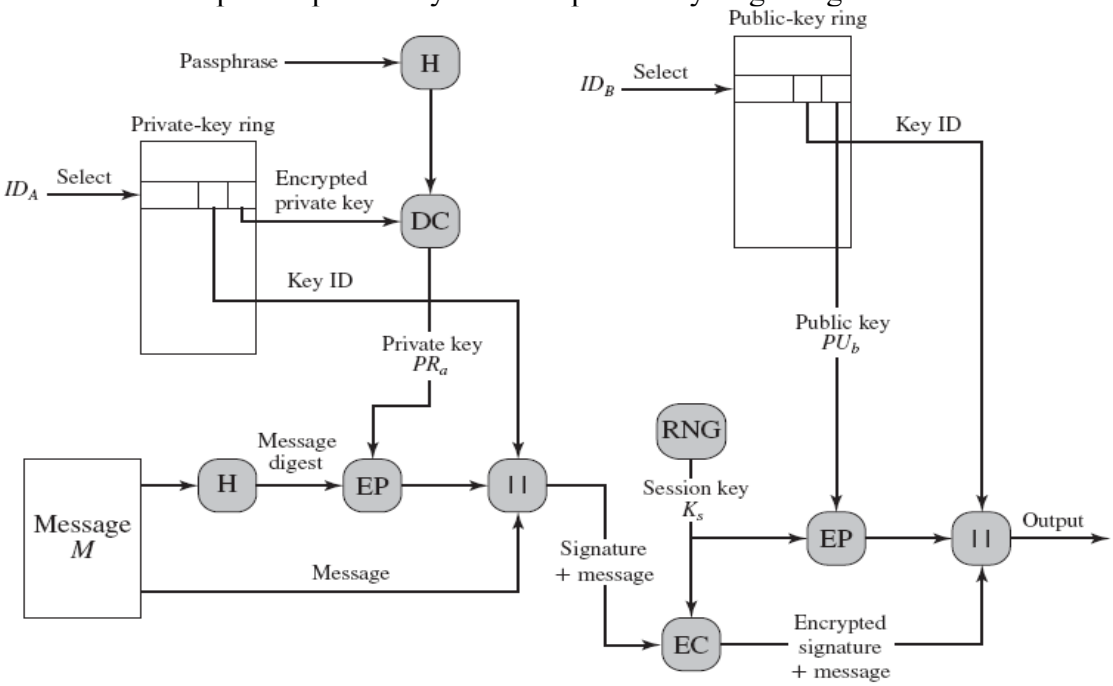
First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

1. Signing the message

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

2. Encrypting the message

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



PGP message generation

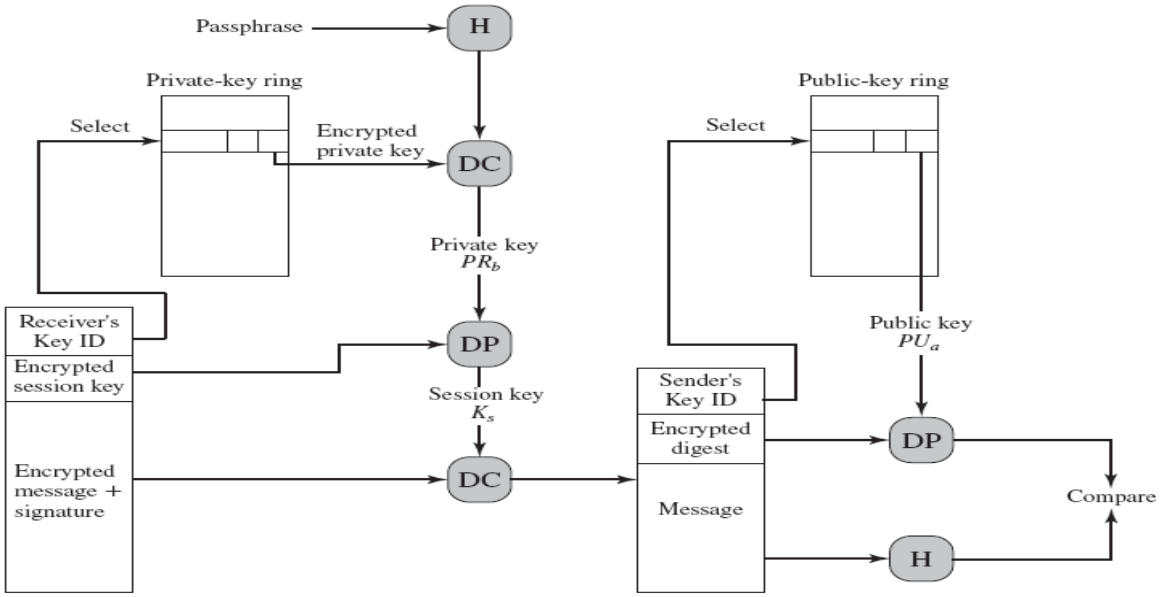
The receiving PGP entity performs the following steps:

1. Decrypting the message

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

2. Authenticating the message

- PGP retrieves the sender's public key from the public key ring, using the key ID field in the signature key component of the message as an index.
- PGP recovers the transmitted message digest.
- PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.



PGP message reception

S/MIME

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

MULTIPURPOSE INTERNET MAIL EXTENSIONS

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and **limitations of the use of Simple Mail Transfer Protocol (SMTP)**

1. SMTP **cannot transmit executable files** or other binary objects.
2. SMTP **cannot transmit text data** that includes national language characters,
3. SMTP servers may **reject mail message over a certain size**.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in **translation problems**.
5. SMTP gateways to X.400 electronic mail networks **cannot handle non textual data** included in X.400 messages.

MIME – HEADER FILES

The five header fields defined in MIME are

- ✓ **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

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

Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript.
	octet-stream	General binary data consisting of 8-bit bytes.

MIME TRANSFER ENCODINGS

Table 18.4 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.

S/MIME FUNCTIONALITY

- ✓ **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.
- ✓ **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.

CRYPTOGRAPHIC ALGORITHMS

- ✓ Create message digest to form digital signature
  - Must use SHA-1, Should support MD5
- ✓ Encrypt message digest to form signature
  - Must support DSS, Should support RSA
- ✓ Encrypt session key for transmission
  - Should support Diffie-Hellman, Must support RSA
- ✓ Encrypt message for transmission with one-time session key
  - Must support triple DES, Should support AES, Should support RC2/40
- ✓ Create a message authentication code
  - Must support HMAC with SHA-1, Should support HMAC with SHA-1

S/MIME – ENHANCED SECURITY SERVICES

- ✓ Signed receipts --The receiver returns a signed receipt back to the sender to verify the message arrived
- ✓ Security labels --Permission, priority or role of message being sent
- ✓ Secure mailing lists--Sending to multiple recipients at once securely by using a public key for the whole mailing list

S/MIME CERTIFICATE PROCESSING

- ✓ S/MIME uses X.509 v3 certificates
- ✓ managed using a hybrid of a strict X.509 CA hierarchy & PGP's web of trust
- ✓ each client has a list of trusted CA's certs
- ✓ and own public/private key pairs & certs
- ✓ certificates must be signed by trusted CA's

S/MIME Messages

Table 2: S/MIMEContent Types			
Type	**Subtype**	**SMIME Parameter**	**Description**
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
//Application	PKCS 7-MIME	Signed Data	A signed S/MIME entity.
	PKCS 7-MIME	Enveloped Data	An encrypted S/MIME entity.
	PKCS 7-MIME	degenerate signed Data	An entity containing only public-key certificates.
	PKCS 7-MIME	Compressed Data	A compressed S/MIME entity
	PKCS 7-SIGNATURE	signed Data	The content type of the signature subpart of a multipart/signed message.

SECURING A MIME ENTITY

- ✓ S/MIME secures a MIME entity with a signature, encryption, or both.
- ✓ A MIME entity may be an entire message or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message.
- ✓ The MIME entity is prepared according to the normal rules for MIME message preparation.

ENVELOPED DATA

The steps for preparing an envelopedData MIME entity are

1. **Generate a pseudorandom session key** for a particular symmetric encryption algorithm (RC2/40 or triple DES).
2. For each recipient, **encrypt the session key** with the recipient's public RSA key.
3. For each recipient, **prepare a block known as RecipientInfo** that contains an identifier of the recipient's public-key certificate, for an identifier of the algorithm used to encrypt the session key, and the encrypted session key.
4. **Encrypt the message content with the session key**.

The RecipientInfo blocks followed by the encrypted content constitute the envelopedData. This information is then encoded into base64.

SIGNED DATA

The signedData smime-type can be used with one or more signers. The steps for preparing a signedData MIME entity are

1. **Select a message digest algorithm** (SHA or MD5).
2. **Compute the message digest** (hash function) of the content to be signed.
3. **Encrypt the message** digest with the **signer's private key**.
4. **Prepare a block known as SignerInfo** that contains the signer's public key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

- ✓ The signedData entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and SignerInfo. The signedData entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64.

Clear Signing

- Achieved using the multipart content type with a signed subtype
- This signing process does not involve transforming the message to be signed
- Recipients with MIME capability but not S/MIME capability are able to read the incoming message

REGISTRATION REQUEST

The certification request includes certification RequestInfo block, followed by an identifier of the public-key encryption algorithm, followed by the signature of the certificationRequestInfo block made using the sender's private key.

The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key.

CERTIFICATES-ONLY MESSAGE

A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request. The message is an application/pkcs7-mime type/subtype with ansmime-type parameter of degenerate. The steps involved are the same as those for creating a signedData message, except that there is no message content and the signerInfo field is empty.

USER AGENT ROLE

An S/MIME user has several key-management functions to perform.

- ✓ **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating separate Diffie-Hellman and DSS key pairs and SHOULD be capable of generating RSA key pairs. Each key pair MUST be generated from a good source of nondeterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.
- ✓ **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.
- ✓ **Certificate storage and retrieval:** User requires access in order to verify incoming signatures and outgoing messages. Such list is maintained by the user or some administrator.

IP SECURITY

OVERVIEW OF IPSEC

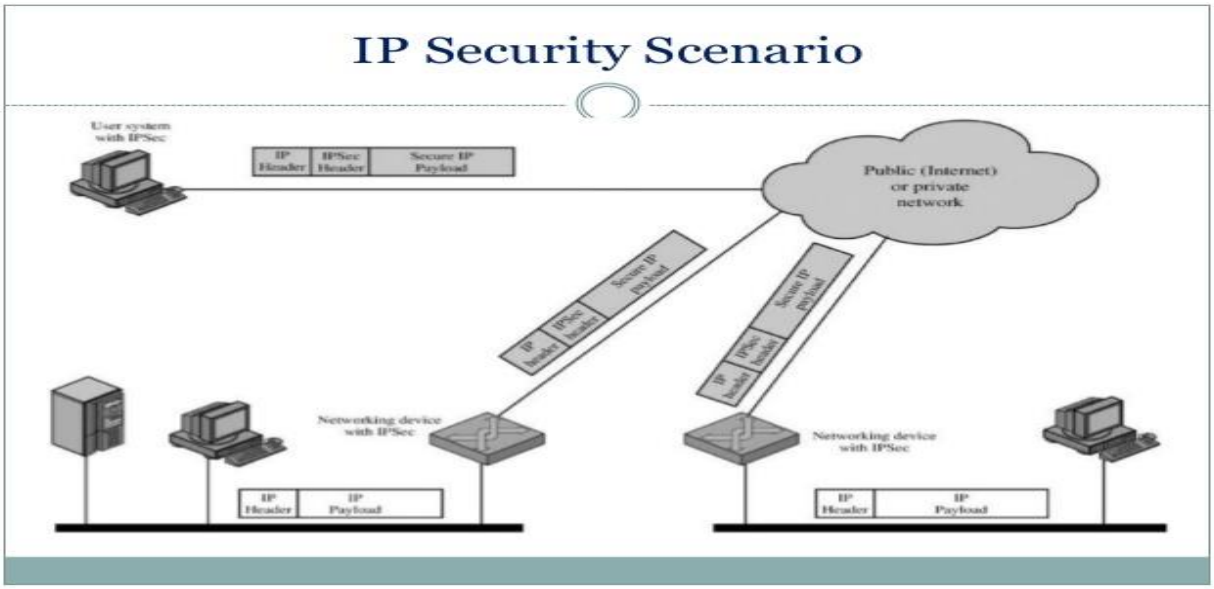
Applications of IPSec

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

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

Benefits of IPSec:

- When IPSec is implemented in a firewall or router, it provides strong security
- IPSec in a firewall is resistant to bypass if all traffic from the outside must use IP, and the firewall is the only means of entrance from the Internet into the organization.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- IPSec can be transparent to end users. There is no need to train users on security mechanisms
- IPSec can provide security for individual users if needed.



Routing Applications

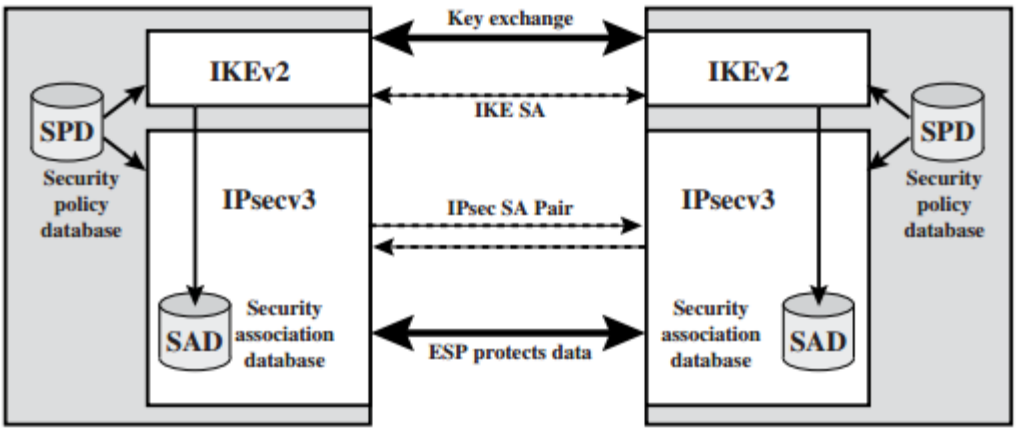
IPSec can play a vital role in the routing architecture required for internet working.

The following are examples of the use of IPSec. IPSec can assure that

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

ARCHITECTURE OF IP SECURITY

- ✓ Fundamental to the operation of IPsec is the concept of a security policy applied to each IP packet that transits from a source to a destination.
- ✓ IPsec policy is determined primarily by the interaction of two databases, the security association database (SAD) and the security policy database (SPD).
- ✓ Security Associations: A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA).
- ✓ An association is a one-way logical connection between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed for two-way secure exchange, then two security associations are required.
- ✓ Security services are afforded to an SA for the use of AH or ESP, but not both.



IP Sec Architecture

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:** 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 field from the outer IP header indicates whether the association is an AH or ESP security association. Hence, in any IP packet, the security association is uniquely identified by the Destination Address in the IPv4 or IPv6 header and the SPI in the enclosed extension header (AH or ESP).

SECURITY ASSOCIATION DATABASE

A security association is normally defined by the following parameters in an SAD entry.

- ✓ **Security Parameter Index**: A 32-bit value selected by the receiving end of an SA to uniquely identify the SA. SPI is used to construct the packet's AH or ESP header.
- ✓ **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.
- ✓ **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.
- ✓ **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with.
- ✓ **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.
- ✓ **IPsec Protocol Mode:** Tunnel, transport, or wildcard.
- ✓ **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging.

SECURITY POLICY DATABASE

The means by which IP traffic is related to specific SAs (or no SA in the case of traffic allowed to bypass IPsec) is the nominal Security Policy Database (SPD).

The following selectors determine an SPD entry:

- ✓ **Remote IP Address:** This may be a single IP address, an enumerated list or range of addresses, or a wildcard (mask) address. The latter two are required to support more than one destination system sharing the same SA (e.g., behind a firewall).
- ✓ **Local IP Address:** This may be a single IP address, an enumerated list or range of addresses, or a wildcard (mask) address. The latter two are required to support more than one source system sharing the same SA (e.g., behind a firewall).
- ✓ **Next Layer Protocol:** The IP protocol header (IPv4, IPv6, or IPv6 Extension) includes a field (Protocol for IPv4, Next Header for IPv6 or IPv6 Extension) that designates the protocol operating over IP. This is an individual protocol number, ANY, or for IPv6 only, OPAQUE. If AH or ESP is used, then this IP protocol header immediately proceeds the AH or ESP header in the packet.
- ✓ **Name:** A user identifier from the operating system. This is not a field in the IP or upper-layer headers but is available if IPsec is running on the same operating system as the user.

- ✓ Local and Remote Ports: These may be individual TCP or UDP port values, an enumerated list of ports, or a wildcard port.

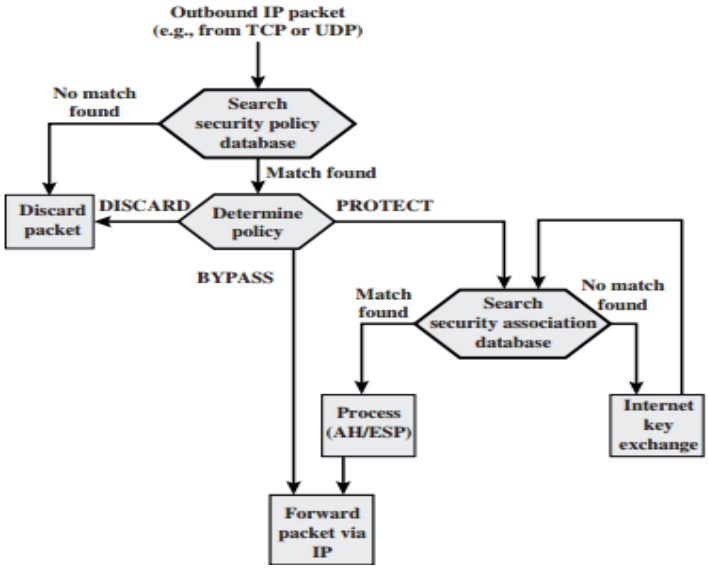
IP TRAFFIC PROCESSING

IPsec is executed on a packet-by-packet basis. When IPsec is implemented, each outbound IP packet is processed by the IPsec logic before transmission, and each inbound packet is processed by the IPsec logic after reception and before passing the packet contents on to the next higher layer.

OUTBOUND PACKETS

A block of data from a higher layer, such as TCP, is passed down to the IP layer and an IP packet is formed, consisting of an IP header and an IP body. Then the following steps occur:

- ✓ IPsec searches the SPD for a match to this packet.
- ✓ If no match is found, then the packet is discarded and an error message is generated.
- ✓ If a match is found, further processing is determined by the first matching entry in the SPD. If the policy for this packet is DISCARD, then the packet is discarded. If the policy is BYPASS, then there is no further IPsec processing; the packet is forwarded to the network for transmission.
- ✓ If the policy is PROTECT, then a search is made of the SAD for a matching entry. If no entry is found, then IKE is invoked to create an SA with the appropriate keys and an entry is made in the SA.
- ✓ The matching entry in the SAD determines the processing for this packet. Either encryption, authentication, or both can be performed, and either transport or tunnel mode can be used. The packet is then forwarded to the network for transmission.

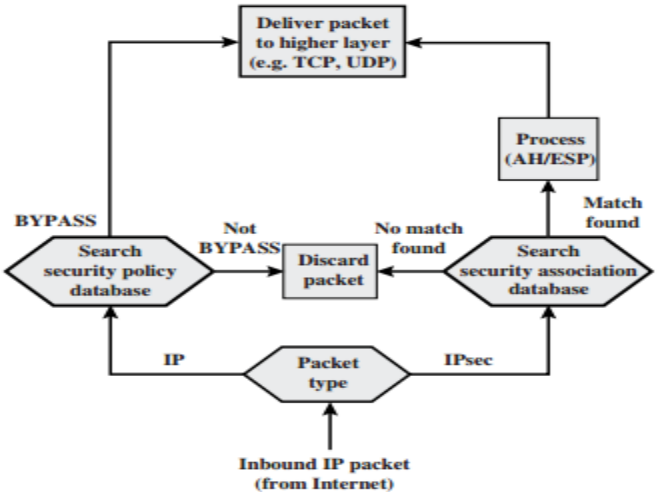


Processing Model for Outbound Packets

INBOUND PACKETS

An incoming IP packet triggers the IPsec processing. The following steps occur:

1. IPsec determines whether this is an unsecured IP packet or one that has ESP or AH headers/trailers, by examining the IP Protocol field (IPv4) or Next Header field (IPv6).
2. If the packet is unsecured, IPsec searches the SPD for a match to this packet. If the first matching entry has a policy of BYPASS, the IP header is processed and stripped off and the packet body is delivered to the next higher layer, such as TCP. If the first matching entry has a policy of PROTECT or DISCARD, or if there is no matching entry, the packet is discarded.
3. For a secured packet, IPsec searches the SAD. If no match is found, the packet is discarded. Otherwise, IPsec applies the appropriate ESP or AH processing. Then, the IP header is processed and stripped off and the packet body is delivered to the next higher layer, such as TCP.



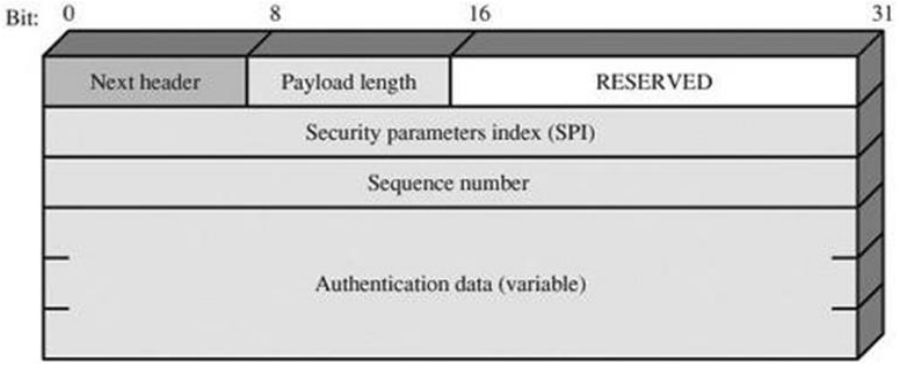
IP Security Header:

1)Authentication Header(AH)

2) Encapsulating Security Payload(ESP)

Authentication Header

- The Authentication Header provides **support for data integrity and authentication of IP packets.**
- The Authentication Header consists of the following fields:
  - **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.
  - **Reserved (16 bits):** For future use.
  - **Security Parameters Index (32 bits):** Identifies a security association.
  - **Sequence Number (32 bits):** A monotonically increasing counter value. This can recognize replayed packets and discard them.
  - **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 produced by MAC algorithm.



IPSec Authentication Header

Anti-Replay Service

A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits it to the intended destination.

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.

If anti-replay is enabled (the default), the sender must not allow the sequence number to cycle past 2^{32}-1 back to zero. Otherwise, there would be multiple valid packets with the same sequence number. If the limit of 2^{32}-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key.

Encapsulating Security Payload

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. **It allows for encryption and integrity protection**.

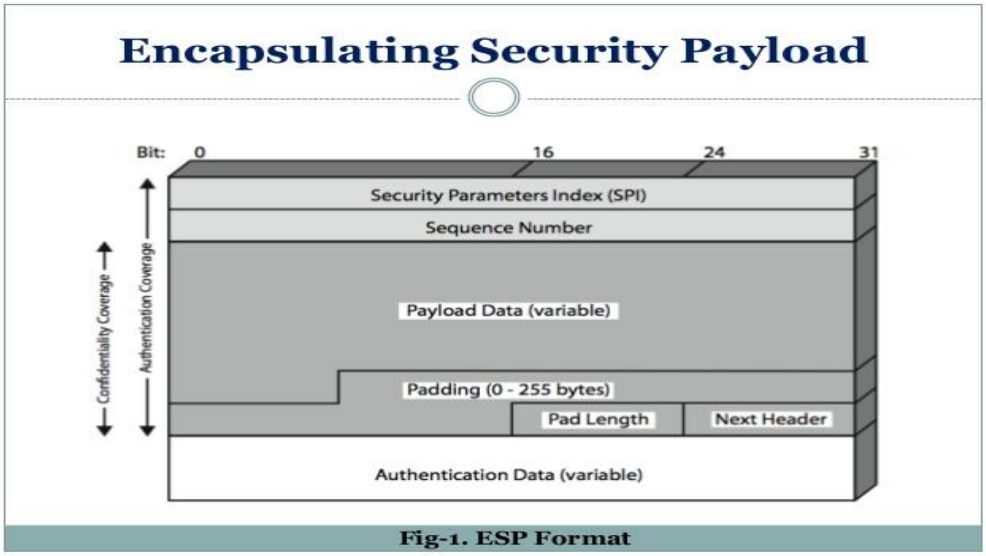


Fig-1. ESP Format

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

- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
- **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
- **Padding (0-255 bytes):** is used for various purposes. Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.
- **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.

Modes of Transfer

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

Transport Mode:

Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet.

Tunnel Mode:

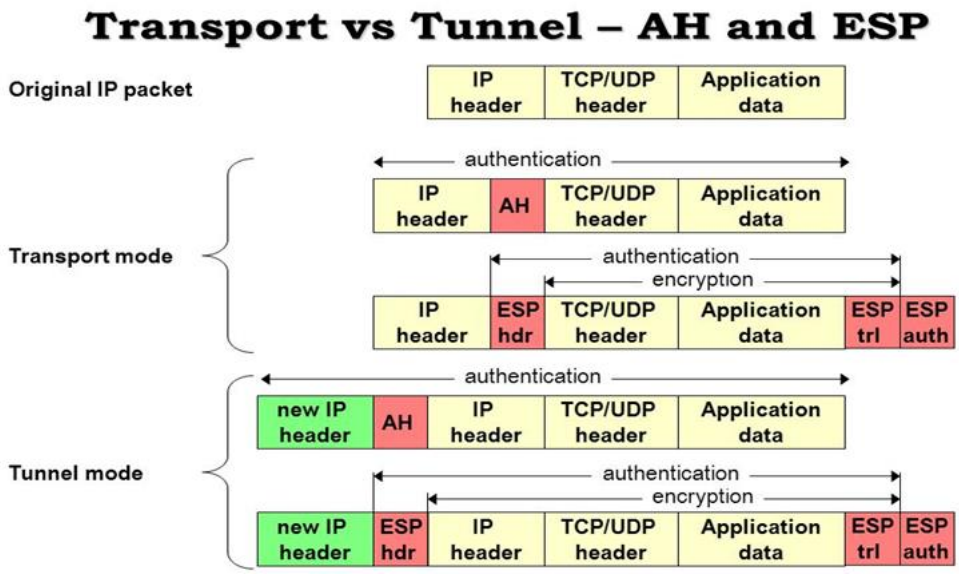
Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header.

The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.

AH & ESP IN IPV4:

For transport mode AH using IPv4, the AH is inserted after the original IP header and before the IP payload

For tunnel mode AH, the entire original IP packet is authenticated, and the AH is inserted between the original IP header and a new outer IP header



AH & ESP USING IPV6

Transport Mode:

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 cipher text plus the ESP header.

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.

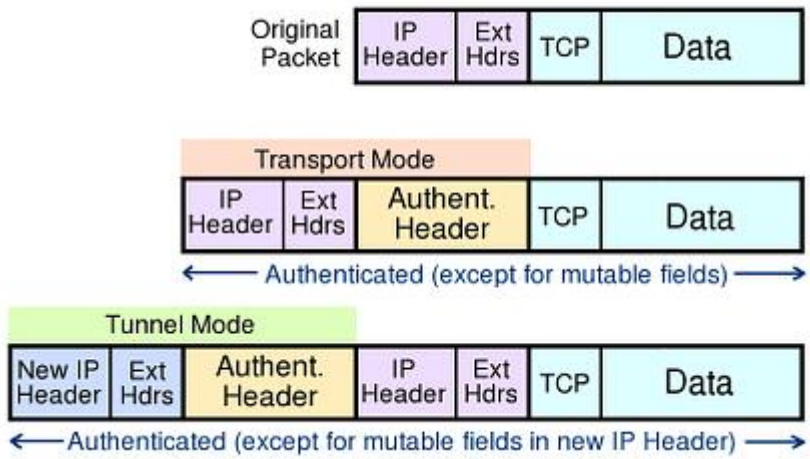
Tunnel Mode ESP:

Tunnel mode ESP is used to encrypt an entire IP packet. For this mode, the ESP header is prefixed to the packet and then the packet plus the ESP trailer is encrypted. This method can be used to counter traffic analysis.

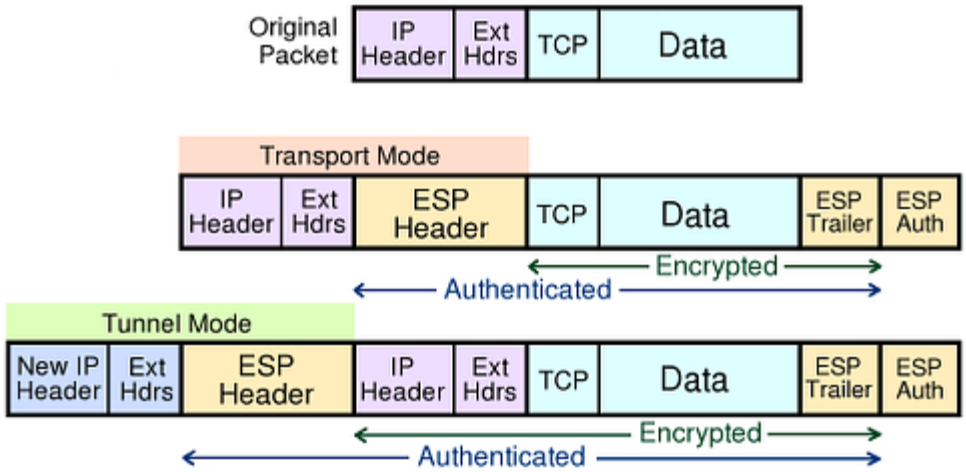
Because the IP header contains the destination address and possibly source routing directives and hop-by-hop option information, it is not possible simply to transmit the encrypted IP packet prefixed by the ESP header. Intermediate routers would be unable to process such a packet.

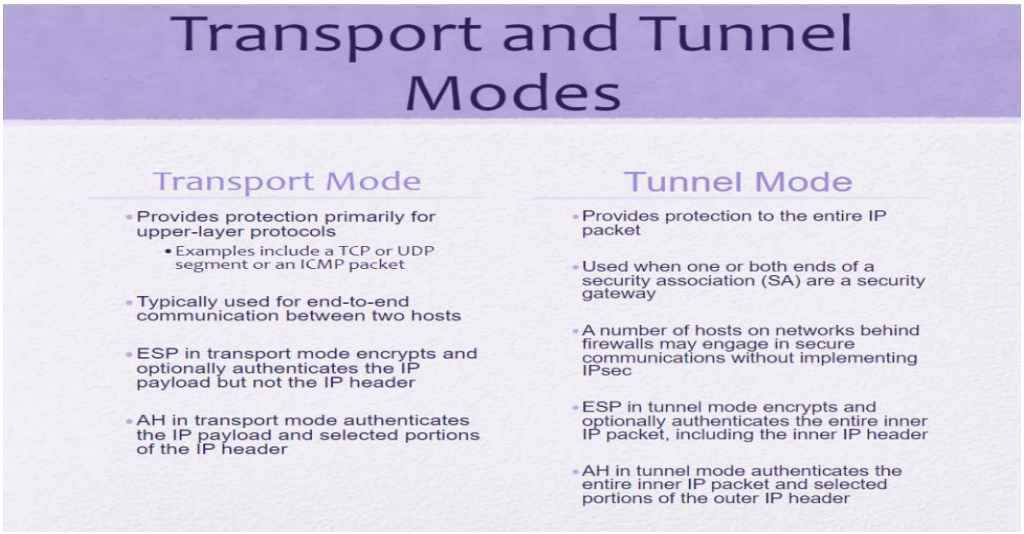
Therefore, it is necessary to encapsulate the entire block (ESP header plus cipher text plus Authentication Data, if present) with a new IP header that will contain sufficient information for routing but not for traffic analysis.

AH FORMAT IN IPV6:



ESP FORMAT IN IPV6:





Internet Key Exchange (Phases of IKE, ISAKMP/IKE Encoding)

- is the protocol used to set up a secure, authenticated communications channel between two parties.
- IKE typically uses X. 509 PKI certificates for authentication and the Diffie–Hellman key exchange protocol to set up a shared session secret.

8 phase 1 IKE protocols :

1. **Public Signature Keys, Main Mode**
2. **Public Signature Keys, Aggressive Mode**
3. **Public Encryption Key, Main Mode, Original**
4. **Public Encryption Key, Aggressive Mode, Original**
5. **Public Encryption Key, Main Mode, Revised**
6. **Public Encryption Key, Aggressive Mode, Revised**
7. **Shared Secret Key, Main Mode**
8. **Shared Secret Key, Aggressive Mode**

In the first message, Alice transmits her "cookie" value. After that, all messages start with the cookie pair (initiator cookie, responder cookie), and that pair serves as the IKE connection identifier.

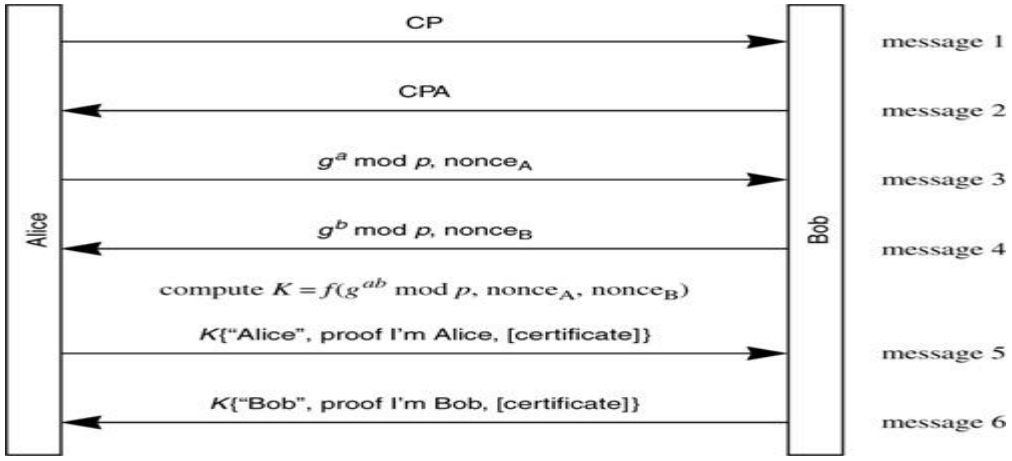
- ✓ Note that in an IKE exchange between Alice and Bob, all messages start with the same cookie pair, in the same order.
- ✓ If Alice initiated the IKE connection, her cookie value always appears in the "initiator cookie" field.

- ✓ To reduce clutter, we won't write "(initiator cookie, responder cookie)" in the figures for the messages. To reduce clutter, **CP indicates crypto proposal, and CPA indicates crypto proposal accepted.**

1. Public Signature Keys, Main Mode

- ✓ In this mode, the two parties have public keys capable of doing signatures. Both endpoint identifiers are hidden from an eavesdropper, but an active attacker can figure out the initiator's identity.
- ✓ The reason for including nonces in messages 3 and 4 is so that Alice and/or Bob can save themselves computation by using the same Diffie-Hellman private value for many exchanges.
- ✓ If they *always* use the same value, then there will not be perfect forward secrecy, so it's a good idea to change it periodically.

Figure: Public signature keys, main mode



2. Public Signature Keys, Aggressive Mode

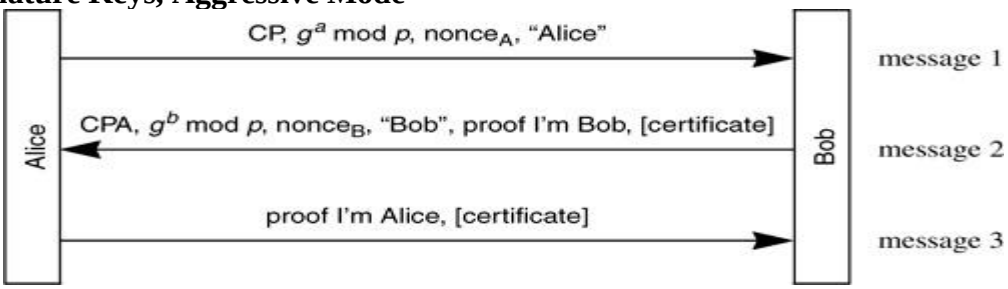


Figure: Public signature keys, aggressive mode

Note that messages 2 and 3 are not encrypted, even though the same information is encrypted in the main mode public signature key variant. The identities could have been encrypted and have the exchange still be 3 messages

3. Public Encryption Key, Main Mode, Original

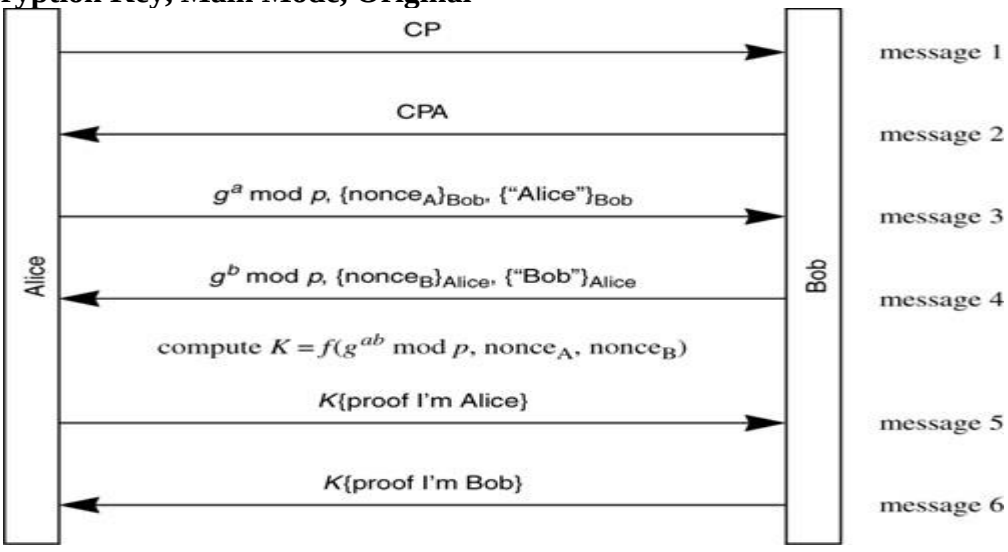


Figure: Public Encryption Keys, main mode, original protocol

- ✓ IKE specifies 4 different phase-1 protocols for public encryption keys, because the original protocols (main mode and aggressive mode) were inefficient (separately encrypted multiple fields with public

keys, requiring multiple private key operations). It's astonishing that they left the original protocols in the spec once they redesigned them.

- ✓ A problem with this variant is that in message 3 there are two fields separately encrypted with Bob's public key, so he needs to do two private key operations to decrypt it. Likewise Alice needs to do two private key operations to decrypt message 4.

4. Public Encryption Key, Aggressive Mode, Original

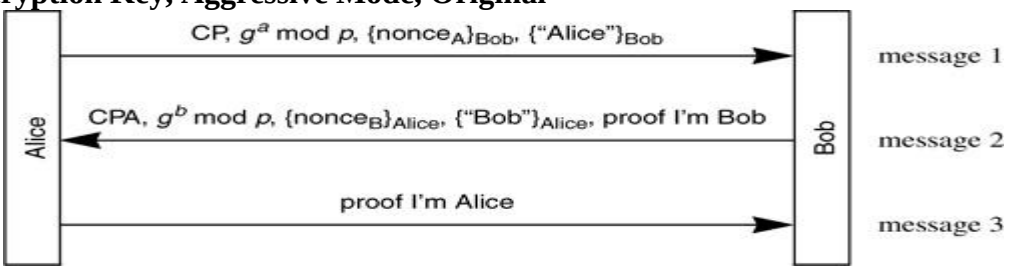


Figure: Public Encryption Keys, aggressive mode, original protocol

- ✓ This protocol is almost the same as the main mode version except that messages 1 and 2 are removed (and crypto suites other than Diffie-Hellman group are negotiated in parallel with the other information in messages 1 and 2) and Bob provides his proof in message 2 rather than, as in main mode, doing it after Alice presents her proof.
- ✓ The proof consists of a hash of the nonce presented by the other side (which requires knowledge of the private key to decrypt), along with the Diffie-Hellman values and the cookie values.

5. Public Encryption Key, Main Mode, Revised

The public encryption protocol was revised to require only a single private key operation on each side (rather than two in the original). This is done by encrypting with a secret key which is a function of the nonce, and the nonce is encrypted with the other side's public key. Thus the other side uses its private key to retrieve the nonce, but then decrypts the other fields with a secret key.

The revised protocol allows Alice to optionally send Bob her certificate. It still has the problem that Alice needs to know Bob's public key.

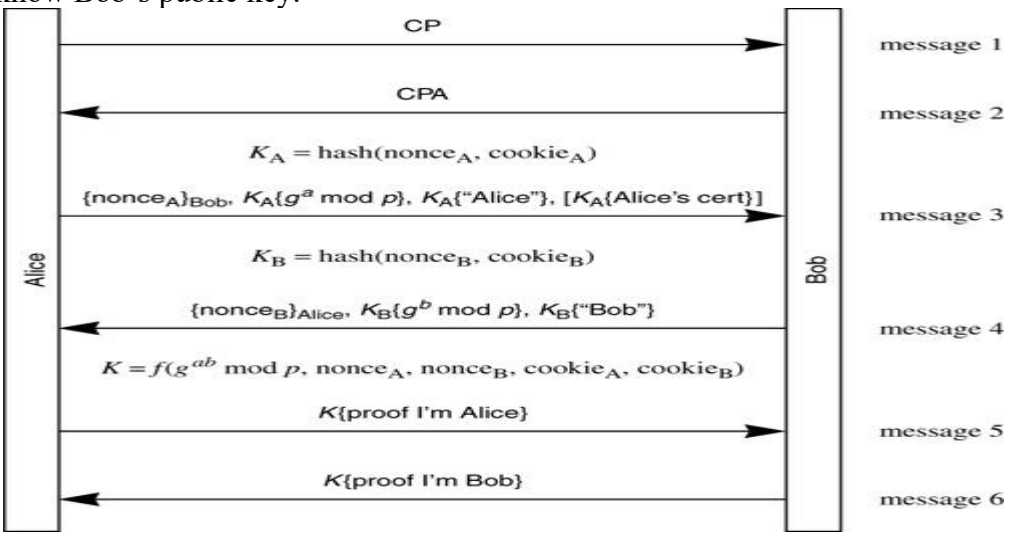


Figure: Public Encryption Keys, main mode, revised protocol

6. Public Encryption Key, Aggressive Mode, Revised

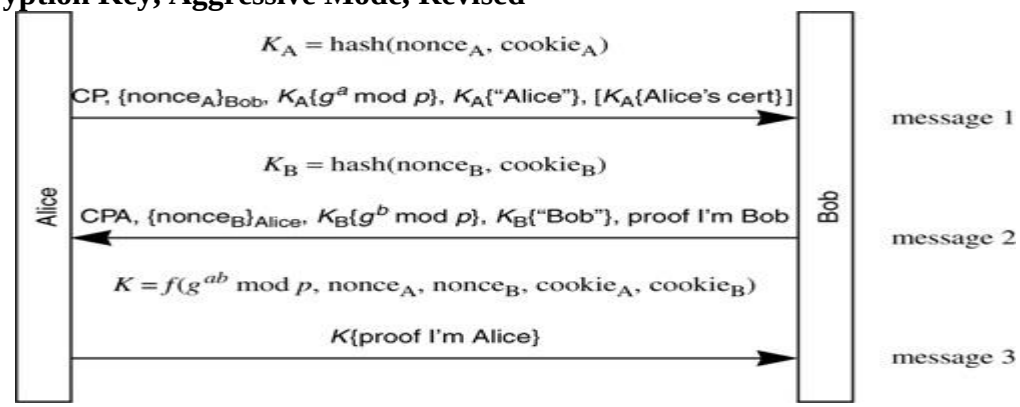


Figure : Public Encryption Keys, aggressive mode, revised protocol

7. Shared Secret Key, Main Mode

- ✓ This is the one required protocol. And it is the most broken. One situation in which this protocol might be useful is in the "road warrior" case, where an employee's laptop is configured with a shared secret with the company's firewall.
- ✓ This would allow the employee to authenticate to the firewall and establish an encrypted tunnel. But the way this mode is designed requires the identities to be the IP addresses. This makes it useless in the road warrior case, because a road warrior's IP address is dependent on where she is that day.
- ✓ The problem with this protocol is that Alice sends her identity in message 5 encrypted with a key K which is a function of the shared secret J. Bob can't decrypt message 5 in order to find out who he's talking to unless he knows J, which means he needs to know who he's talking to. The working group noticed this, and rather than fixing the protocol (which wouldn't have been hard), they instead said that in this mode Alice's identity has to be her IP address! This makes it almost useless in practice, and it certainly doesn't hide identities.

8. Shared Secret Key, Aggressive Mode

This protocol doesn't have the problem that the main mode shared secret protocol has, because the identities are not sent encrypted.

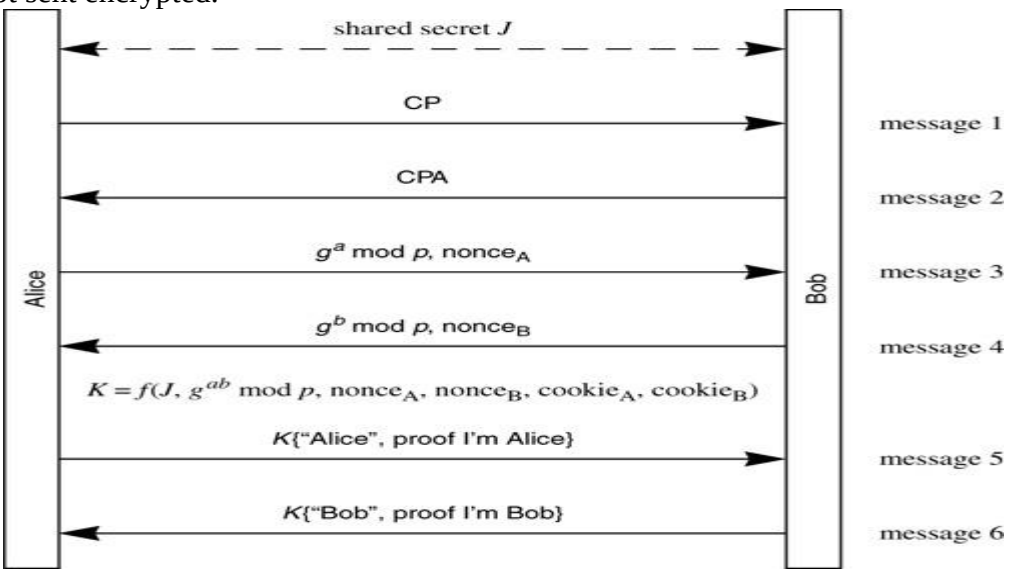


Figure: Pre-shared secret, main mode

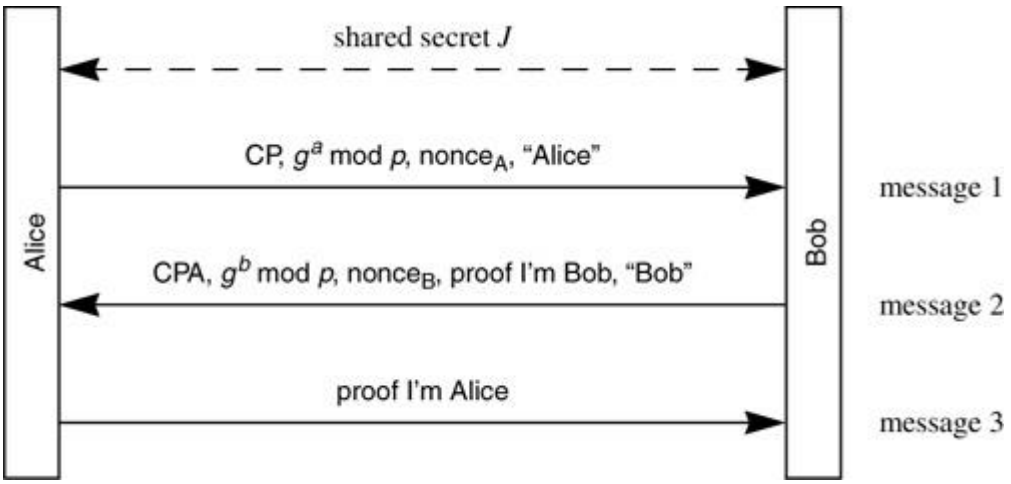
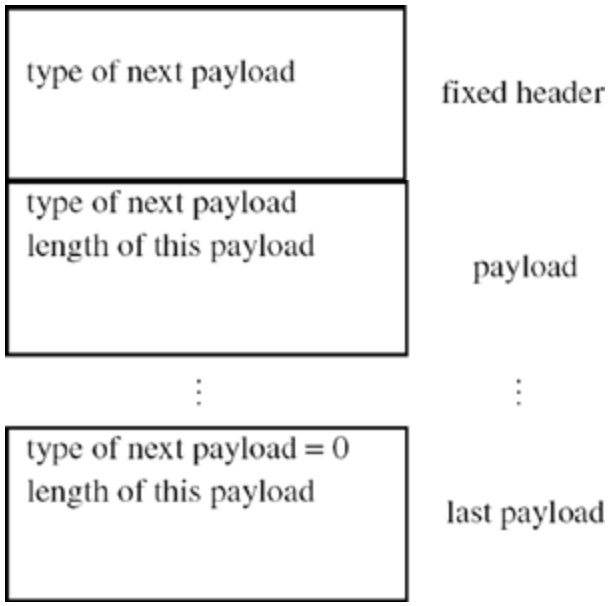


Figure: Pre-shared secret, aggressive mode

ISAKMP/IKE encoding

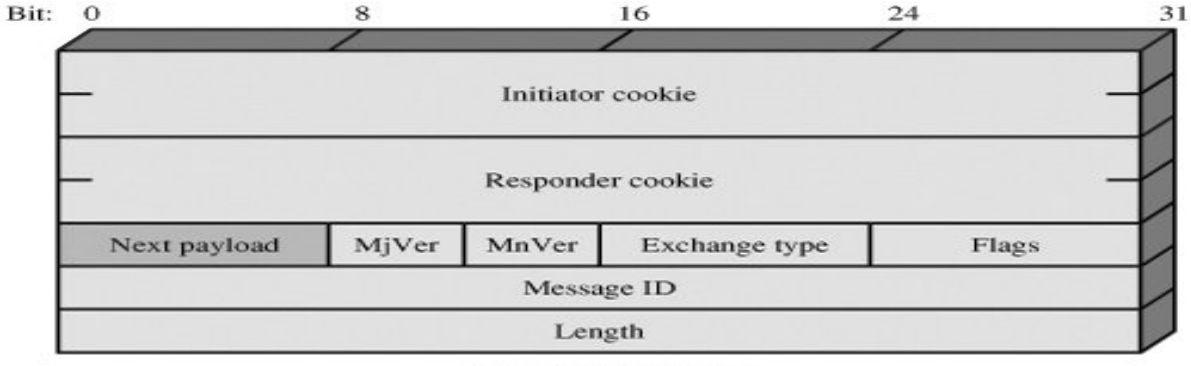
- ✓ Messages have a fixed header, and then a sequence of what ISAKMP refers to as *payloads*. Similar in spirit to IPv6 extension headers, each payload starts with TYPE OF NEXT PAYLOAD and LENGTH OF THIS PAYLOAD.



The payload types are:

- ✓ 0 = end (i.e., no next payload)
- ✓ 1 = SA (security association): contains DOI and "situation", a modifier of DOI, and must include payloads 2 and 3
- ✓ 2 = P (proposal): proposed SPI, or SPI in reverse direction
- ✓ 3 = T (transform): cryptographic choices
- ✓ 4 = KE (key exchange):, the Diffie-Hellman value
- ✓ 5 = ID (endpoint identifier in phase 1, traffic selector in phase 2)
- ✓ 6 = CERT (certificate)
- ✓ 7 = CR (certificate request) (can include the name of the certifier from whom you'd like a certificate)
- ✓ 8 = hash (some sort of checksum)
- ✓ 9 = signature
- ✓ 10 = nonce
- ✓ 11 = notification
- ✓ 12 = delete (subtype of notification, meaning you are closing this SPI)
- ✓ 13 = vendor ID (can be thrown in to show what implementation you're using). To avoid dealing with a registry of vendor IDs, and allowing the field to be fixed size, this is an MD of some sort of string guaranteed to uniquely describe the vendor, such as its name and telephone number.
- ✓ 14–127 reserved for future use
- ✓ 128–255 = private use (i.e., so NSA can use it and not publish what they're using it for)

ISAKMP FORMAT



(a) ISAKMP header

(b) Generic payload header

Fixed Header

All messages start with a 28-octet fixed length header.

# octets		
8	initiator's cookie	
8	responder's cookie	
1	next payload	
1	version number (major/minor)	
1	exchange type	
1	Flags	
4	message ID	
4	message length (in units of octets)	(after encryption)

The fields are:

- ✓ initiator's cookie (8 octets)
- ✓ responder's cookie (8 octets). Note this will =0 in the first message, since it is unknown at that point
- ✓ next payload type
- ✓ version (1 octet). This is worth ranting about. The version number field is divided into two 4-bit fields. The intention is that the top nibble is the major version number and the bottom nibble is the minor version number.
- ✓ exchange type (1 octet). The values defined are:
  - 1 = base. An exchange type defined by ISAKMP but not used by IKE. This adds an extra message to aggressive mode, so that Alice (the initiator) can send her proposed parameters before sending her Diffie-Hellman value, so that the Diffie-Hellman group could also be negotiated.
  - 2 = identity protection. This is what is called "main mode" in IKE.
  - 3 = authentication only. Not used by IKE.
  - 4 = aggressive. Same as what's called "aggressive mode" in IKE.
  - 5 = informational. Not really an "exchange", since it's a single message without an acknowledgment, used to tell the other side something such as that you are refusing the connection because you don't like the version number.
  - 6–31 = reserved values by ISAKMP for assignment by IANA as new ISAKMP exchange types
  - 32–239 = to be defined within a particular DOI
  - 240–255 = for private use
- ✓ flags:
  - bit 0 (LSB): encrypted—whether the fields after the header are encrypted
  - bit 1: commit—A flag so badly named, and so confusingly defined in ISAKMP, that IKE wound up using the same bit and the same name for almost the opposite purpose.
  - bit 2: authentication only—this means that the fields after the header are not encrypted. This bit gives no additional information over merely not setting the "encrypted" flag.
- ✓ Message ID: Unique ID for this message.
- ✓ Message length: Length of entire message, in units of octets.

Payload Portion of ISAKMP Messages

After the fixed header comes a set of ISAKMP "payloads", reminiscent of IPv6 "next headers". Each one starts with four octets consisting of:.

# octets	
1	type of next payload
1	reserved (unused, set to zero)
2	length of this payload

The encoding would be more intuitive to have each payload indicate the type of that payload rather than the following one, but this way works too. It's this way because it looks more like IPv6.

SA Payload

- ✓ Assembly of SA payload requires great peace of mind. The SA payload for IKE includes the P (proposal) and T (transform) payloads. The encoding is extremely confusing for no good reason.
- ✓ The SA, P and T each look like independent payloads, but ISAKMP defines Ts as being carried inside a P, and Ps carried inside an SA payload. For example, if you have an SA payload that includes 2 proposals, the first of which includes 4 transforms, and the second of which includes 2 transforms, you'd have the payloads SA, P, T, T, T, T, P, T, T.

Payload Length in SA, P, and T Payloads

- ✓ **The PAYLOAD LENGTH in the SA payload is the length of the entire set of the payloads consisting of the SA and all Ps and Ts associated with that SA.** The payload length of each P is the length of that P payload plus the T payloads that follow. The payload length of each T payload is actually the length of that T payload.

WEB SECURITY CONSIDERATIONS

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets.

Web Security Threats

A Comparison of Threats on the Web

	Threats	Consequences	Countermeasures
Integrity	Modification of user data Trojan horse browser Modification of memory Modification of message traffic in transit	Loss of information Compromise of machine 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 requests Filling up disk or memory Isolating machine by DNS attacks	Disruptive Annoying Prevent user from getting work done	Difficult to prevent
Authentication	Impersonation of legitimate users Data forgery	Misrepresentation of user Belief that false information is valid	Cryptographic techniques

SECURE SOCKET LAYER AND TRANSPORT LAYER SECURITY

SSL:

SSL is designed to make use of TCP to provide a reliable end-to-end secure service.

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.

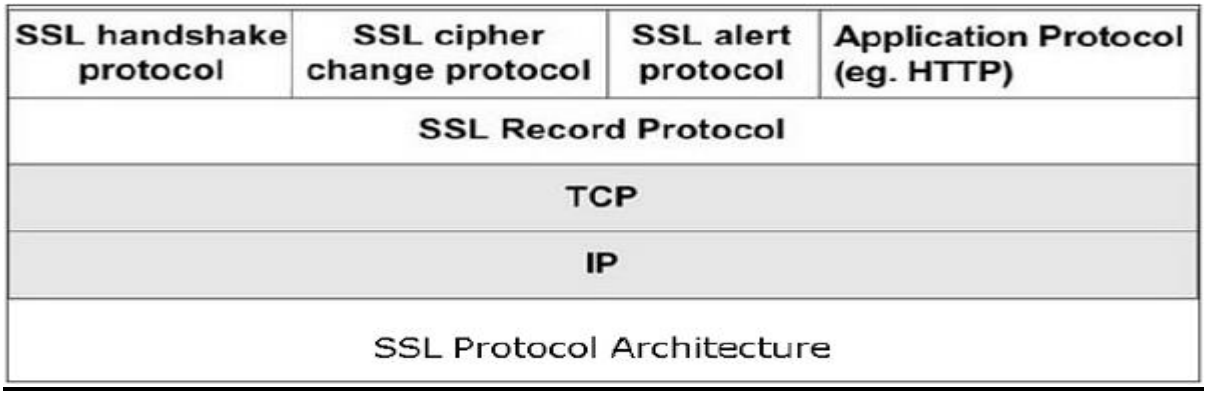
A session state is defined by the following parameters

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
- Master secret
- Is resumable

A connection state is defined by the following parameters:

- Server and client random
- Server write MAC secret
- Client write MAC secret
- Server write key
- Client write key.
- Initialization vectors
- Sequence numbers

SSL Architecture:



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

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

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 diagram indicates the overall operation of the SSL Record Protocol. 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 first step is fragmentation. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

The next step in processing is to compute a **message authentication code** over the compressed data.

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 $2^{14} + 2048$.

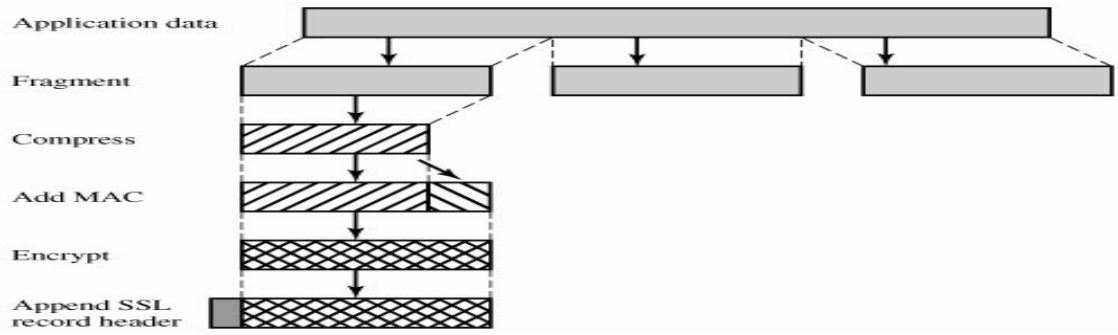


Fig: SSL Record Protocol Operation

- ChangeCipherSpec Protocol

✓ Simplest part of SSL protocol. It comprises of a single message exchanged between two communicating entities, the client and the server.

✓ As each entity sends the ChangeCipherSpec message, it changes its side of the connection into the secure state as agreed upon.

✓ The cipher parameters pending state is copied into the current state.

✓ Exchange of this Message indicates all future data exchanges are encrypted and integrity is protected.

- **SSL Alert Protocol**

✓ This protocol consists of a single message which consists of a single byte with the value **Alert Protocol**

✓ The Alert Protocol is used to convey SSL-related alerts to the peer entity.

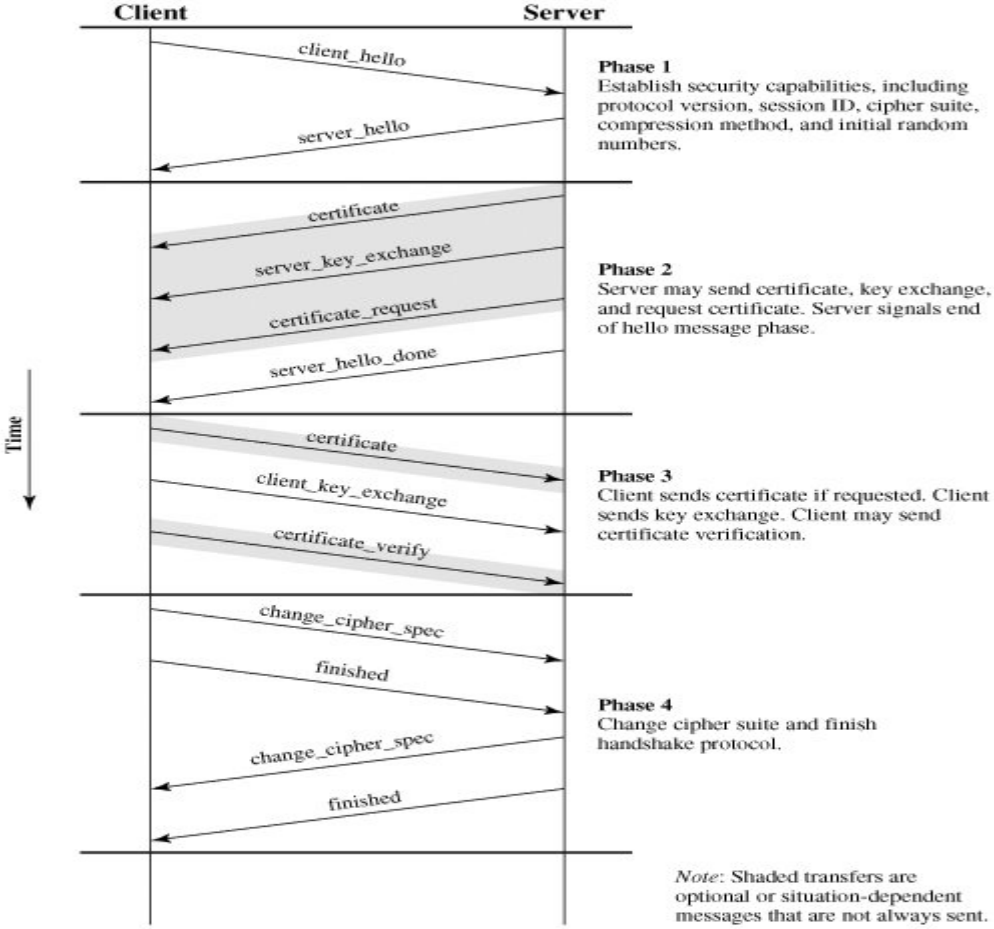
✓ Each message in this protocol consists of two bytes.

✓ The first byte takes the value **warning(1) or fatal(2)** to convey the severity of the message.

✓ The second byte contains a code that indicates the specific alert. This protocol is used to report errors – such as unexpected message, bad record MAC, security parameters negotiation failed, etc.

✓ It is also used for other purposes – such as notify closure of the TCP connection, notify receipt of bad or unknown certificate, etc.

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

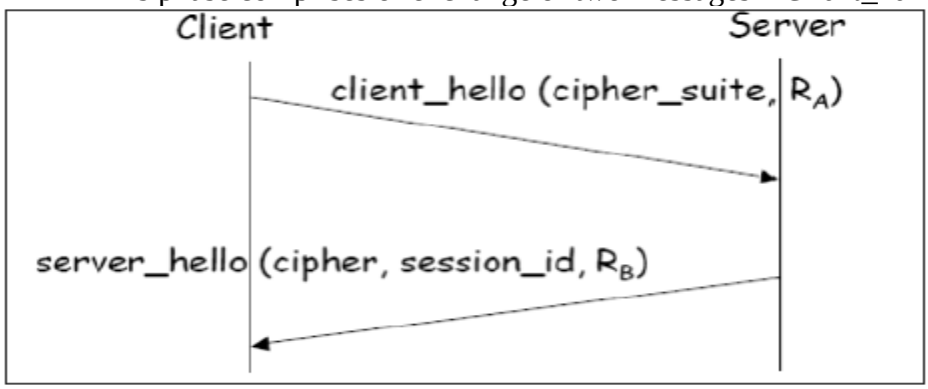


Establishment of SSL Session

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

Phase 1 − Establishing security capabilities.

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



- *Client_hello* contains of list of cryptographic algorithms supported by the client, in decreasing order of preference.

- *Server_hello* contains the selected Cipher Specification (CipherSpec) and a new *session_id*.
- The CipherSpec contains fields like −
    - ✓ Cipher Algorithm (DES, 3DES, RC2, and RC4)
    - ✓ MAC Algorithm (based on MD5, SHA-1)
    - ✓ Public-key algorithm (RSA)
    - ✓ Both messages have "nonce" to prevent replay attack.

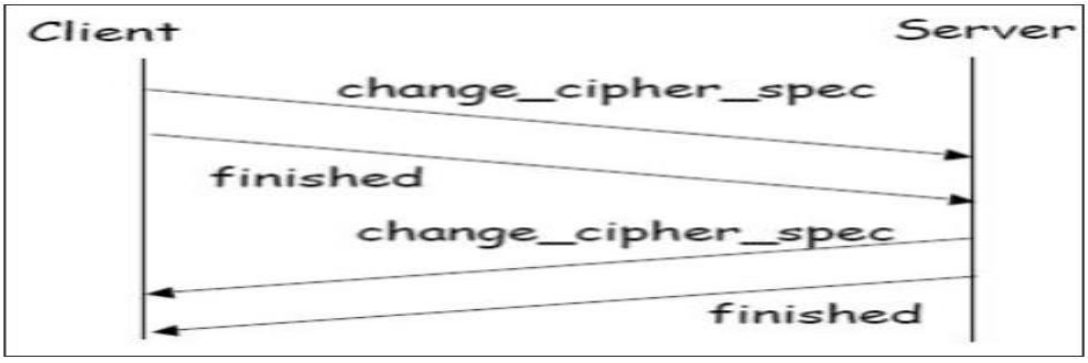
Phase 2 − Server authentication and key exchange.

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

Phase 3 − Client authentication and key exchange.

- ✓ Client sends certificate, only if requested by the server.
- ✓ It also sends the Pre-master Secret (PMS) encrypted with the server's public key.
- ✓ Client also sends *Certificate_verify* message if certificate is sent by him to prove he has the private key associated with this certificate. Basically, the client signs a hash of the previous messages.

Phase 4 − Finish.



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

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

SSL Session Keys

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

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

TLS Protocol

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

Salient Features

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

Comparison of TLS and SSL Protocols

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

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

2)Message Authentication − TLS employs a keyed-hash message authentication code (H-MAC). Benefit is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.

3)Session Key Generation − There are two differences between TLS and SSL protocol for generation of key material. Method of computing pre-master and master secrets is similar. But in TLS protocol, computation of master secret uses the HMAC standard and pseudorandom function (PRF) output instead of ad-hoc MAC. The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.

4)**Alert Protocol Message** −TLS protocol supports all the messages used by the Alert protocol of SSL, except *No certificate* alert message being made redundant. The client sends empty certificate in case client authentication is not required.Many additional Alert messages are included in TLS protocol for other error conditions such as *record_overflow, decode_error* etc.

5)Supported Cipher Suites − SSL supports RSA, Diffie-Hellman and Fortezza cipher suites. TLS protocol supports all suits except Fortezza.

6)Client Certificate Types − TLS defines certificate types to be requested in a *certificate_request* message. SSLv3 support all of these. Additionally, SSL support certain other types of certificate such as Fortezza.

7)CertificateVerify and Finished Messages −In SSL, complex message procedure is used for the *certificate_verify* message. With TLS, the verified information is contained in the handshake messages itself thus avoiding this complex procedure. Finished message is computed in different manners in TLS and SSLv3.

8)Padding of Data − In SSL protocol, the padding added to user data before encryption is the minimum amount required to make the total data-size equal to a multiple of the cipher's block length. In TLS, the padding can be any amount that results in data-size that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

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

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

SET FOR E-COMMERCE TRANSACTIONS

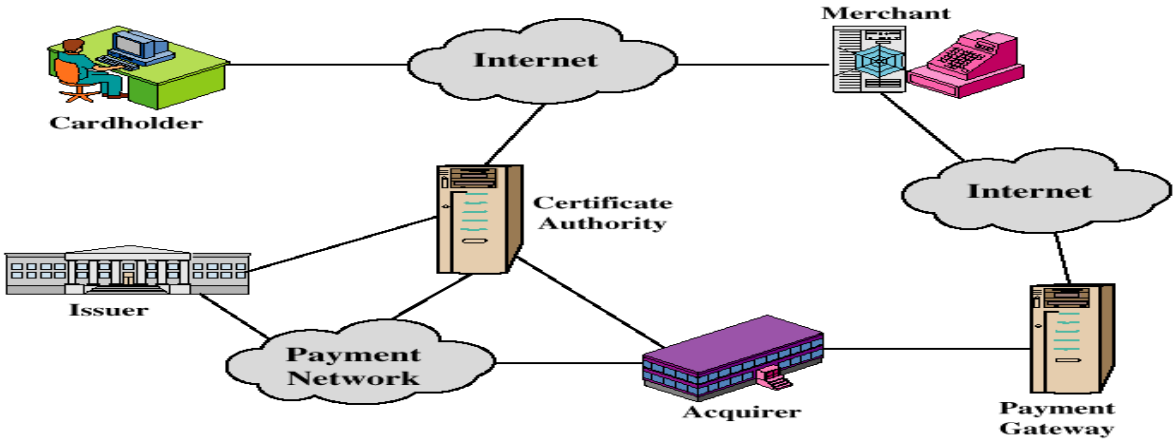
- ✓ The Secure Electronic Transaction (SET) is a protocol designed for protecting credit card transactions over the Internet.
- ✓ It is an industry-backed standard that was formed by MasterCard and Visa (acting as the governing body) in February 1996.
- ✓ SET relies on cryptography and X.509 v3 digital certificates to ensure message confidentiality and security.
- ✓ SET is the only Internet transaction protocol to provide security through authentication. It combats the risk of transaction information being altered in transit by keeping information securely encrypted at all times and by using digital certificates to verify the identity of those accessing payment details.

Business Requirements for SET

1. **Confidentiality of information -** provide confidentiality of payment and order information
2. **Integrity of data -** Ensure the integrity of all transmitted data
3. **Cardholder account authentication -** Provide authentication that a cardholder is a legitimate customer of a branded payment card account
4. **Merchant authentication -** Provide authentication that a merchant can accept credit card transactions through its relationship with an acquiring financial institution
5. **Security techniques -** Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction

6. **Creation of brand-new protocol -** Create a protocol that neither depends on transport security mechanisms nor prevents their use)
7. **Interoperability -** Facilitate and encourage interoperability among software and network providers).
It will be appropriate to introduce the TCP/IP model and Internet Protocol suite, including Electronic Payment System in Figure.

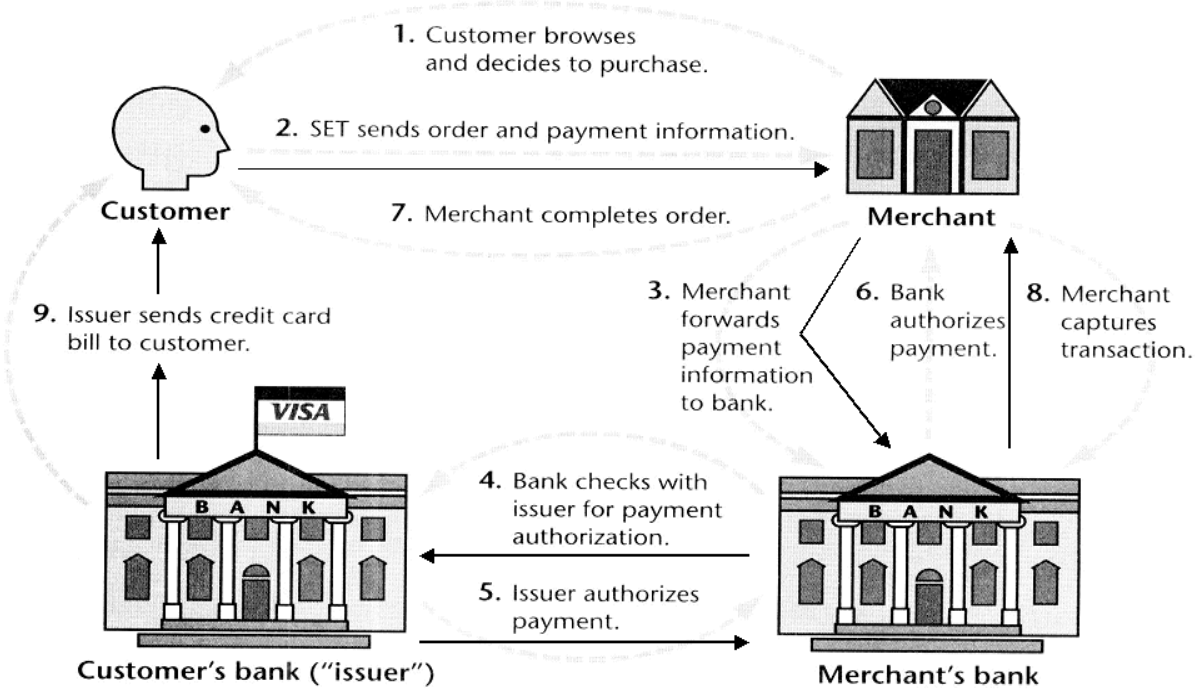
COMPONENTS OF SET:



SET Transactions :

- ✓ The customer opens an account with a card issuer.
  - ✓ MasterCard, Visa, etc.
- ✓ The customer receives a X.509 V3 certificate signed by a bank.
  - ✓ X.509 V3
  - ✓ It verifies the customer's RSA public key and its expiration date.
- ✓ A merchant who accepts a certain brand of card must possess two X.509 V3 certificates.
- ✓ One for signing & one for key exchange
- ✓ The customer places an order for a product or service with a merchant.
- ✓ The merchant sends a copy of its certificate for verification.

These transactions are as shown in the following diagram



SET Transaction

- **Customer opens account:** The customer obtains a credit card account, such as MasterCard or Visa, with a bank that supports electronic payment and SET.
- **Customer receives a certificate:** After verification the customer receives X.509V3, digital certificate which is signed by the bank. This certificate verifies the customer's RSA public key and expiration date.

- **Merchants have their own certificates:**
  - ✓ Merchants who accepts card need to have 2 certificates for 2 public keys owned by them.
  - ✓ One certificate is used for signing of message and the other is used for key exchange.
  - ✓ The merchants also need the copy of payment gateway's public key certificate.
- **Customer places an order:**
  - ✓ The customer places the order containing the list of items to be purchased to the merchant.
  - ✓ The merchant returns the order form having the items, price, total price and order number.
- **Merchant is verified:** The merchant along with the order form sends its certificate copy. The customer can verify the same.
- **Order and payment are sent:**
  - ✓ The customer sends order and payment information into the merchant along with customer's certificate.
  - ✓ This is order conformation of the order form.
  - ✓ The payment contains the card details. This is encrypted, so it cannot be read by the merchant.
  - ✓ The certificate sent can be verified by the merchant.
- **Merchant requests payment authorization:** The merchant sends the payment information to the payment gateway. The merchant requests for authentication of the customer, credit limit, validity.
- **Merchant confirms order:** The merchant sends conformation of the order to the customer.
- **Merchant provides goods or service**
- **Merchant requests payment**

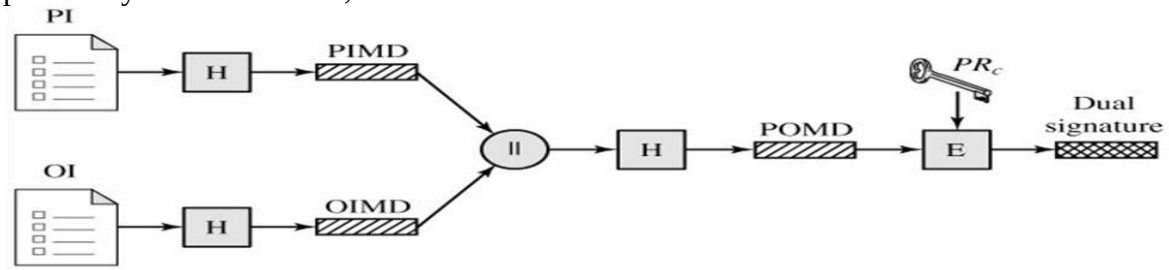
Dual Signature

The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order.

The customer 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

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

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.



PI = Payment information — PIMD = PI message digest
OI = Order information — OIMD = OI message digest
H = Hash function(SHA-1) — POMD = Payment order message digest
II = Concatenation — E = Encryption(RSA)
PR_C = Customer's private signature key

Payment Processing

- Purchase request
- Payment authorization
- Payment capture

Purchase Request

Before the Purchase Request exchange begins, the cardholder has completed browsing, selecting, and ordering. The end of this preliminary phase occurs when the merchant sends a completed order form to the customer.

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

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

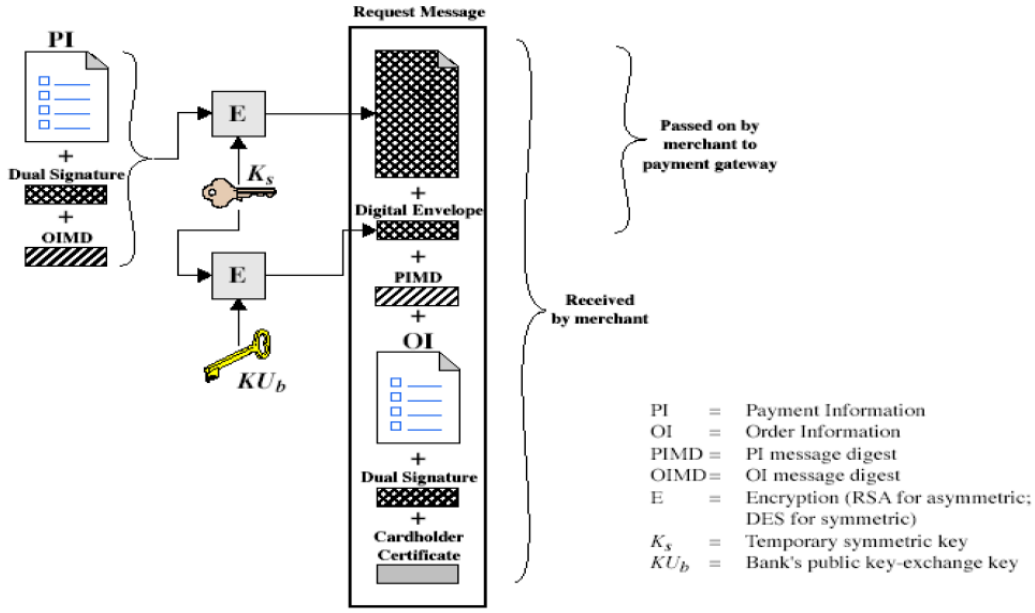


Fig : Purchase Request – Customer

Payment Authorization

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. The payment authorization exchange consists of two messages: Authorization Request and Authorization response.

- Verifies all certificates
- Decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
- Verifies merchant's signature on authorization block
- Decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
- Verifies dual signature on payment block
- Verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
- Requests & receives an authorization from issuer
- Sends authorization response back to merchant

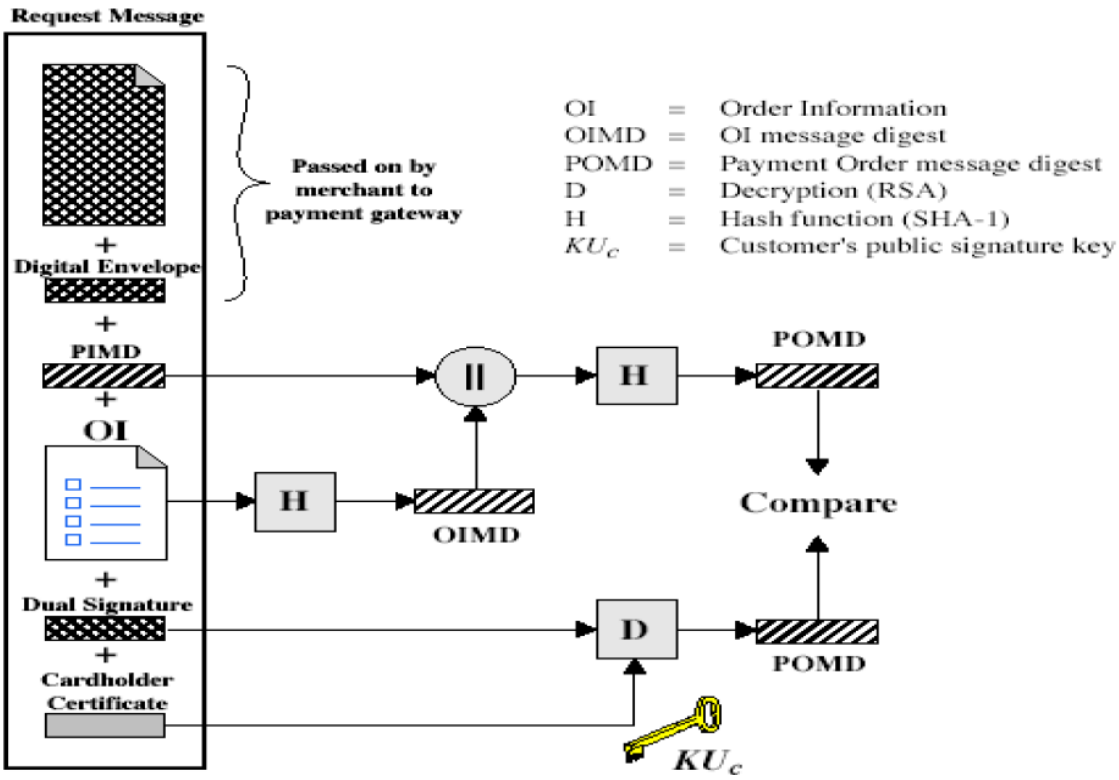


Fig: Purchase Request – Merchant

Payment Capture

To obtain payment, the merchant engages the payment gateway in a payment capture transaction, consisting of a capture request and a capture response message.

- Merchant sends payment gateway a payment capture request
- Gateway checks request
- Then causes funds to be transferred to merchants account
- Notifies merchant using capture response

INTRUDER

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:

1. **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.
2. **Misfeasor** – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.
3. **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.

Intrusion Techniques

The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system.

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

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

The following techniques are used 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.
3. Try words in the system's online dictionary or a list of likely passwords.
4. Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.
5. Try user's phone number, social security numbers and room numbers.
6. Try all legitimate license plate numbers.
7. Use a torjan horse to bypass restriction on access.
8. Tap the line between a remote user and the host system.

Two principle countermeasures:

1. **Detection** – concerned with learning of an attack, either before or after its success.
2. **Prevention** – challenging security goal

INTRUSION DETECTION

Motivation

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 are compromised.
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.

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

Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruder.

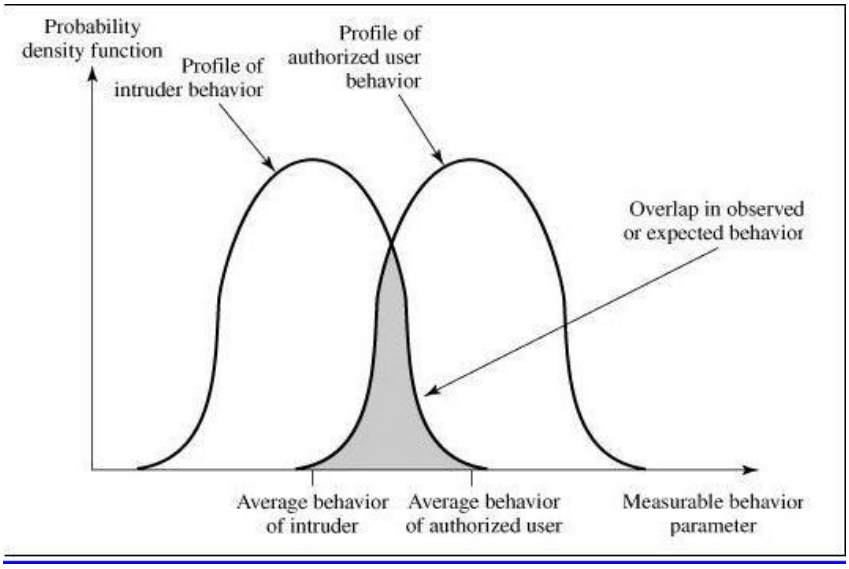


Figure: Profiles of Behavior of Intruders and Authorized Users

Approaches to intrusion detection

Statistical anomaly detection
1)Threshold detection 2)Profile based

Rule-Based detection
1)Anomaly detection 2)Penetration identification

Distributed Intrusion Detection
Host agent module, LAN monitor agent module, Central manager module

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.

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

A **multivariate model** is based on correlations between two or more variables.

A **Markov process model** is used to establish transition probabilities among various states.

A **time series model** focuses on time intervals.

An **operational model** is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records.

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

b) **Profile based:** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
Examples of metrics that are useful for profile-based intrusion detection are the following:

- **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action.
- **Gauge:** A nonnegative integer that may be incremented or decremented.
- **Interval timer**: The length of time between two related events.
- **Resource utilization**: Quantity of resources consumed during a specified period.

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.

a) **Rule-based Anomaly detection:** Rules are developed to detect deviation from previous usage patterns.
   - ✓ Analyze historical audit records to identify usage patterns & auto-generate rules for them
   - ✓ Then observe current behavior & match against rules to see if conforms
   - ✓ Like statistical anomaly detection does not require prior knowledge of security flaws

b) Rule-based penetration identification: An expert system approach that searches for suspicious behavior.

3. Distributed Intrusion Detection

- Traditional focus is on single systems
- But typically have networked systems
- More effective defense has these working together to detect intrusions

Issues

- o Dealing with varying audit record formats
- o Integrity & confidentiality of networked data
- o Centralized or decentralized architecture

Architecture for Distributed Intrusion Detection
Detection Three main components

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

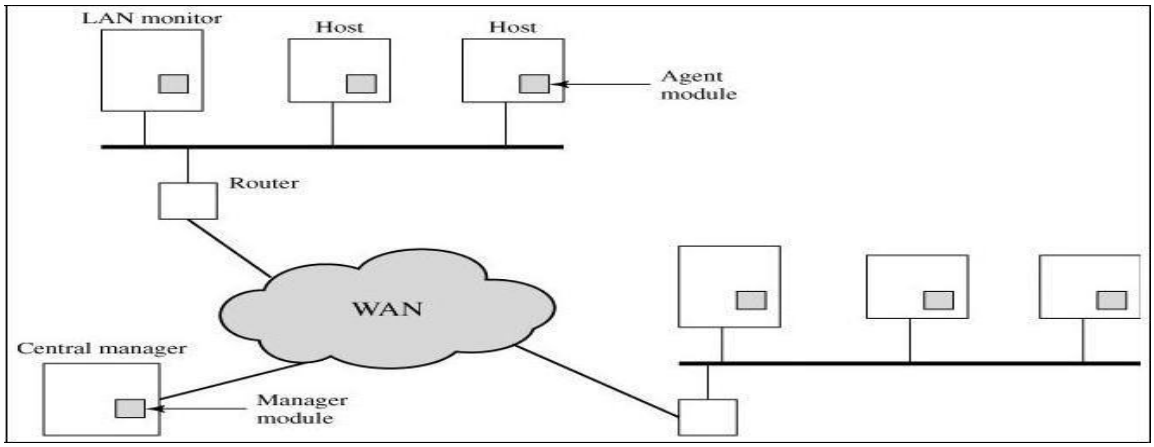


Figure : Architecture for Distributed Intrusion

Honeypots

Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to

- Divert an attacker from accessing critical systems
- Collect information about the attacker's activity
- Encourage the attacker to stay on the system long enough for administrators to respond

Audit Records

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

1. **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.
2. **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Each audit record contains the following fields:

- **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users.
- **Action:** Operation performed by the subject on or with an object; for example, login, read, perform I/O, execute.
- **Object:** Receptors of actions. Examples include files, programs, messages, records,terminals, printers, and user- or program-created structures
- **Exception-Condition:** Denotes which, if any, exception condition is raised on return.
- **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource
- **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

MALICIOUS SOFTWARE

Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.

VIRUS AND RELATED THREATS Malicious Programs

Table: Terminology of 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

Table: Terminology of Malicious Programs	
Name	**Description**

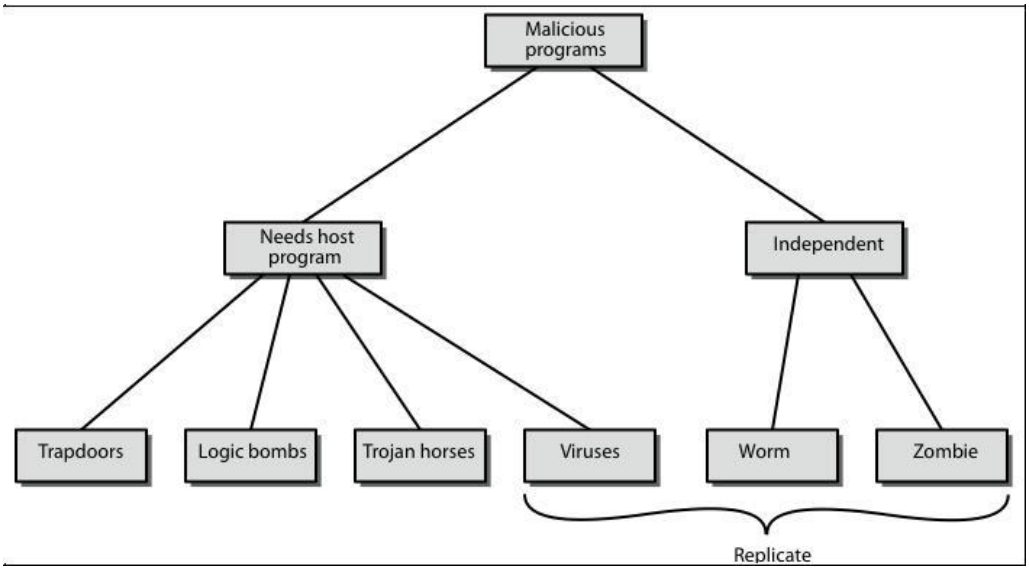
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

Malicious software can be divided into two categories:

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

Programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples.

Programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.



We can also differentiate between those software threats that do not replicate and those that do. A Programs or fragments of programs that are activated by a trigger.

Example: logic bombs, backdoors, and zombie programs.

A program fragment or an independent program that, when executed, may produce one or more copies of itself to be activated later on the same system or some other system.

Example: Viruses and worms are examples.

Backdoor or Trapdoor

- Secret entry point into a program
- Allows those who know access bypassing usual security procedures
- Have been commonly used by developers
- A threat when left in production programs allowing exploited by attackers
- Very hard to block in O/S
- Requires good s/w development & update

Logic Bomb

- One of oldest types of malicious software
- Code embedded in legitimate program
- Activated when specified conditions met
  - Eg presence/absence of some file
  - Particular date/time
  - Particular user
- When triggered typically damage system
  - Modify/delete files/disks, halt machine, etc

Trojan Horse

- Program with hidden side-effects
- Which is usually superficially attractive
  - Eg game, s/w upgrade etc
- When run performs some additional tasks
  - Allows attacker to indirectly gain access they do not have directly
- Often used to propagate a virus/worm or install a backdoor
- Or simply to destroy data

Zombie

- Program which secretly takes over another networked computer
- Then uses it to indirectly launch attacks
- Often used to launch distributed denial of service (ddos) attacks
- Exploits known flaws in network systems

The Nature of Viruses

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

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

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

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

Virus Structure

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

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

- The first line of code is a jump to the main virus program. The second line is a special marker

- That is used by the virus to determine whether or not a potential victim program has already been infected with this virus.
- When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.
- This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.
- Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

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

1. For each uninfected file P_2 that is found, the virus first compresses that file to produce P'_2, which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P'_1, is uncompressed.
4. The uncompressed original program is executed.

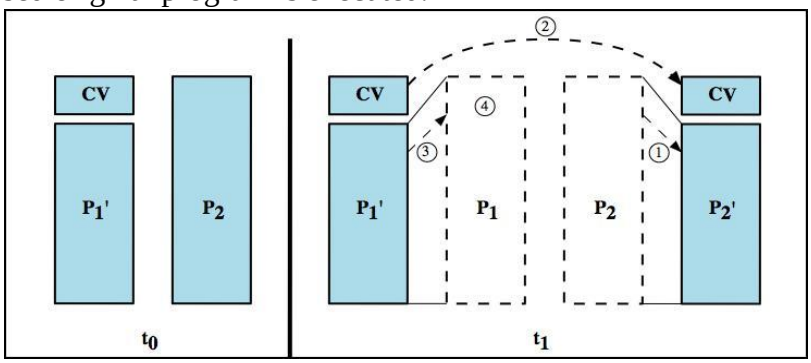


Figure: A Compression Virus

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

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.

Types of Viruses

Following categories as being among the most significant types of viruses:

1. **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.
2. **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.
3. **Boot sector virus:** Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
4. **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software.
5. **Polymorphic virus:** A virus that mutates with every infection, making detection by the "signature" of the virus impossible.
6. **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.

7. **Macro Viruses**
    - 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.
    - 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.
    - 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. In essence, a macro is an executable program embedded in a word processing document or other type of file.
8. **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.

Worms
- Replicating but not infecting program
- Typically spreads over a network
- Using users distributed privileges or by exploiting system vulnerabilities
- Widely used by hackers to create zombie pc's, subsequently used for further attacks
- Major issue is lack of security of permanently connected systems

To replicate itself, a network worm uses some sort of network vehicle. 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.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

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

As with viruses, network worms are difficult to counter.

The Morris Worm
- Best known classic worm
- Released by robert morris in 1988
- Targeted unix systems
- Using several propagation techniques
    - Simple password cracking of local PW file
    - Exploit bug in finger daemon
    - Exploit debug trapdoor in send mail daemon
- If any attack succeeds then replicated self

Recent Worm Attacks
- New spate of attacks from mid-2001
- Code Red - used MS IIS bug
    - probes random IPs for systems running IIS
    - had trigger time for denial-of-service attack
    - 2nd wave infected 360000 servers in 14 hours
- Code Red 2 - installed backdoor
- Nimda - multiple infection mechanisms
- SQL Slammer - attacked MS SQL server
- Sobig.f - attacked open proxy servers
- Mydoom - mass email worm + backdoor

Worm Technology

- **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, and other 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.

VIRUS COUNTERMEASURES

Antivirus Approaches

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

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

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.
There are four generations of antivirus software:

- **First-generation -** simple scanners
  - Scanner uses virus signature to identify virus
  - Or change in length of programs
- **Second-generation -** heuristic scanners
  - Uses heuristic rules to spot viral infection
  - Or uses crypto hash of program to spot changes
- **Third-generation -** activity traps
  - Memory-resident programs identify virus by actions
- **Fourth-generation -** full-featured protection
  - Packages with a variety of antivirus techniques
  - Eg scanning & activity traps, access-controls

Advanced Antivirus Techniques

1. Generic Decryption

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

- **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.
- **Virus signature scanner:** A module that scans the target code looking for known virus signatures.
- **Emulation control module:** Controls the execution of the target code.

At the start of each simulation, the emulator begins interpreting instructions in the target code, one at a time. Thus, if the code includes a decryption routine that decrypts and hence exposes the virus, that code is interpreted. In effect, the virus does the work for the antivirus program by exposing the virus. Periodically, the control module interrupts interpretation to scan the target code for virus signatures.

2. Digital Immune System

- General purpose emulation & virus detection
- Any virus entering org is captured, analyzed, detection/shielding created for it, removed

Error! Hyperlink reference not valid. illustrates the typical steps in digital immune system operation

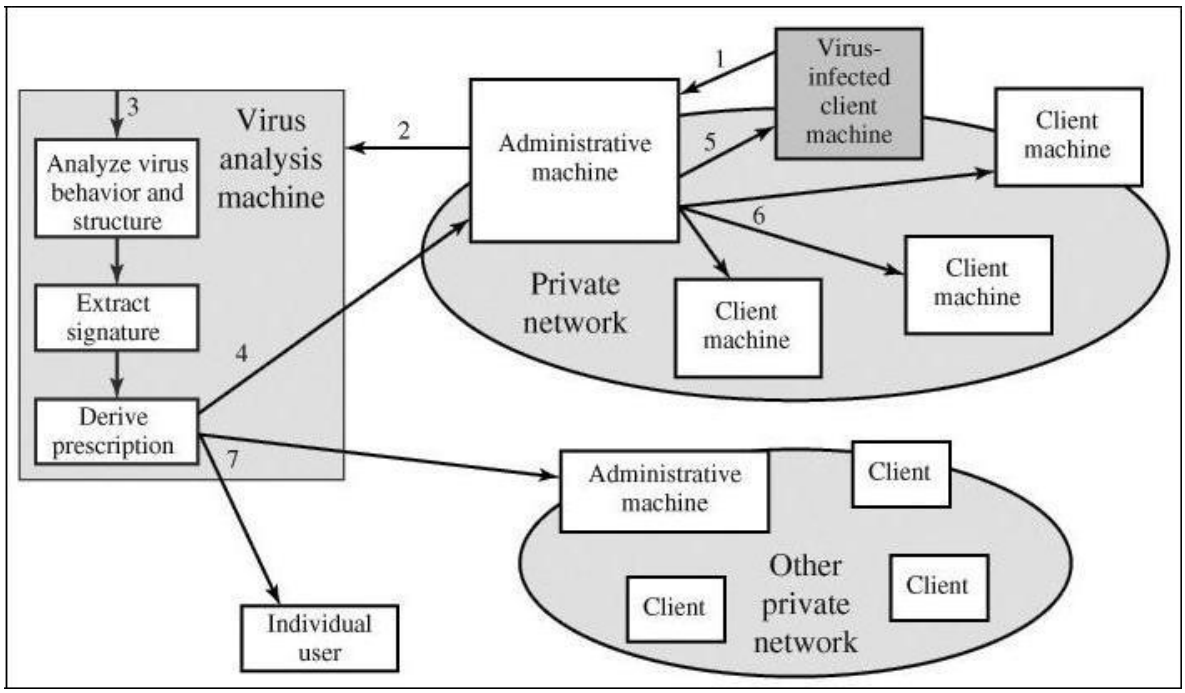


Figure: Digital Immune System

Steps

1. A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.
2. The administrative machine encrypts the sample and sends it to a central virus analysis machine.
3. This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored. The virus analysis machine then produces a prescription for identifying and removing the virus.
4. The resulting prescription is sent back to the administrative machine.
5. The administrative machine forwards the prescription to the infected client.
6. The prescription is also forwarded to other clients in the organization.
7. Subscribers around the world receive regular antivirus updates that protect them from the new virus.

3. Behavior-Blocking Software

- Integrated with host O/S
- Monitors program behavior in real-time
  - Eg file access, disk format, executable mods, system settings changes, network access
- For possibly malicious actions
  - If detected can block, terminate, or seek ok
- Has advantage over scanners
- But malicious code runs before detection

Monitored behaviors can include the following:

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

FIREWALLS DESIGN PRINCIPLES

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets.

This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach.

The alternative, increasingly accepted, is the firewall. The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter.

The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed.

The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Firewall Characteristics

- All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies.
- The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

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

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

Capabilities of Firewall

- A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
- A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.
- A firewall is a convenient platform for several internet functions that are not security related.
- A firewall can serve as the platform for IPsec.

Limitations of Firewall

- The firewall cannot protect against attacks that bypass the firewall.
- The firewall does not protect against internal threats.
- 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.

INTERNET FIREWALLS FOR TRUSTED SYSTEMS

- A firewall is a device or group of devices that controls access between networks.
- A firewall generally consists of filters and gateway(s), varying from firewall to firewall.
- It is a security gateway that controls access between the public Internet and an intranet and is a secure computer system placed between a trusted network and an untrusted Internet.
- A firewall is an agent that screens network traffic in some way, blocking traffic it believes to be inappropriate, dangerous, or both.
- Firewalls act as an intermediate server in handling SMTP and HTTP connections in either direction.

ROLES OF FIREWALLS

- The firewall imposes restrictions on packets entering or leaving the private network.
- All traffic from inside to outside, and vice versa, must pass through the firewall, but only authorized traffic will be allowed to pass.
- Packets are not allowed through unless they conform to a filtering specification or unless there is negotiation involving some sort of authentication.
- The firewall itself must be immune to penetration.
- Firewalls create checkpoints between an internal private network and an untrusted Internet. Once the choke points established, the device can monitor, filter, and verify all inbound and outbound traffic.
- The firewall may filter on the basis of IP source and destination addresses and TCP port number.
- The firewall also enforces logging and provides alarm capacities as well. Placing logging services at firewalls, security administrators can monitor all access to and from the Internet.
- Firewalls may block TELNET or RLOGIN connections from the Internet to the intranet. It block SMTP and FTP connections to the Internet from internal systems not authorized to send e-mail or to move files.
- The firewall provides protection from various kinds of IP spoofing and routing attacks. It can also serve as the platform for IPsec.
- A firewall can limit network exposure by hiding the internal network systems and information from the public Internet.
- A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.

Four general techniques that firewalls use to control access and enforce the site's security policy

1. **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.

2. **Direction control:** Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.
3. **User control:** Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users).
4. **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.

Firewall limitations

- It cannot protect against internal threats such as an employee who cooperates with an external attacker.
- The firewall cannot protect against attacks that bypass the firewall.
- It is also unable to protect against the transfer of virus-infected programs or files because it is impossible for it to scan all incoming files, e-mail, and messages for viruses.

FIREWALL-RELATED TERMINOLOGY

To design and configure a firewall, some familiarity with the basic terminology is required.

1. Bastion Host

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

Bastion hosts must check all incoming and outgoing traffic and enforce the rules specified in the security policy. They must be prepared for attacks from external and possibly internal sources.

The bastion host's role falls into the following three common types:

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

2. **Dual-homed bastion host:** This is a firewall device with at least two network interfaces. The advantage of using such hosts is that they create a complete break between the external network and the internal network. This break forces all incoming and outgoing traffic to pass through the host.
3. **Multihomed bastion host:** Used to allow the user to enforce strict security mechanisms. When the security policy requires all inbound and outbound traffic to be sent through a proxy server, a new proxy server should be created for the new streaming application.

2. Proxy Server

- Proxy servers are used to communicate with external servers on behalf of internal clients.
- A proxy service is set up and torn down in response to a client request, rather than existing on a static basis.
- Application proxies forward packets only when a connection has been established using some known protocol. When the connection closes, a firewall using application proxies rejects individual packets, even if they contain port numbers allowed by a rule set.
- The audit log is an essential tool for detecting and terminating intruder attacks. Therefore, each proxy maintains detailed audit information by logging all traffic, each connection, and the duration of each connection.

- Each proxy is independent of other proxies on the bastion host. If there is a problem with the operation of any proxy, or if future vulnerability is discovered, it is easy to replace the proxy without affecting the operation of the proxy's applications.
- A proxy generally performs no disk access other than to read its initial configuration file. This makes it difficult for an intruder to install Trojan horse sniffers or other dangerous files on the bastion host.

3. SOCKS

- The SOCKS protocol version 4 provides for unsecured firewall traversal for TCP-based client/server applications, including HTTP, TELNET, and FTP.
- The new protocol extends the SOCKS version 4 model to include UDP, allows the framework to include provision for generalized strong authentication schemes, and extends the addressing scheme to encompass domain name and IPv6addresses.
- SOCKS define how to establish authenticated connections, but currently, it does not provide a clear-cut solution to the problem of encrypting the data traffic.

4. Choke Point

- A choke point is the point at which a public Internet can access the internal network.
- The most comprehensive and extensive monitoring tools should be configured on the choke points.
- Proper implementation requires that all traffic be funneled through these choke points.
- All traffic is flowing through the firewalls, security administrators, as a firewall strategy, need to create choke points to limit external access to their networks.
- Once these choke points have been clearly established, the firewall devices can monitor, filter, and verify all inbound and outbound traffic.

5. Demilitarized Zone (DMZ)

- The DMZ is an expression that originates from the Korean War. It meant a strip of land forcibly kept clear of enemy soldiers.
- In terms of a firewall, the DMZ is a network that lies between an internal private network and the external public network.
- DMZ networks are sometimes called *perimeter networks*.
- A DMZ is used as an additional buffer to further separate the public network from the internal network.
- A gateway is a machine that provides relay services to compensate for the effects of a filter.
- The network inhabited by the gateway is often called the *DMZ*.
- A gateway in the DMZ is sometimes assisted by an internal gateway.

6. Logging and Alarms

- Logging is usually implemented at every device in the firewall, but these individual logs combine to become the entire record of user activity.
- Since a choke point is installed at the firewall, a prospective hacker will go through the choke point.
- If so, the comprehensive logging devices will probably capture all hacker activities, including all user activities as well.
- The user can then tell exactly what a hacker is doing and have such information available for audit.
- The audit log is an essential tool for detecting and terminating intruder attacks.
- Many firewalls allow the user to preconfigure responses to unacceptable activities.

- The firewall should alert the user by several means.
- The two most common actions are for the firewall to break the TCP/IP connection or to have it automatically set off alarms.

7. VPN

- VPNs are appropriate for any organization requiring secure external access to internal resources.
- All VPNs are tunnelling protocols in the sense that their information packets or payloads are encapsulated or tunnelled into the network packets.
- All data transmitted over a VPN is usually encrypted because an opponent with access to the Internet could eavesdrop on the data as it travels over the public network.
- The VPN encapsulates all the encrypted data within an IP packet.
- Authentication, message integrity, and encryption are very important fundamentals for implementing a VPN.
- Without such authentication procedures, a hacker could impersonate anyone and then gain access to the network.

TYPES OF FIREWALLS

Firewalls are classified into three common types:

1. **Packet filtering Router**
2. **Circuit-level gateways**
3. **Application-level gateways**

1. Packet filtering Router

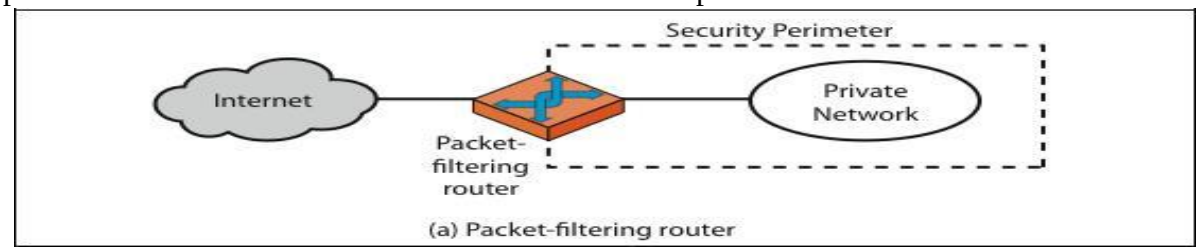
A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

Source IP address – IP address of the system that originated the IP packet. **Destination IP address** – IP address of the system, the IP is trying to reach.

Source and destination transport level address – transport level port number.

IP protocol field – defines the transport protocol.

Interface – for a router with three or more ports, which interface of the router the packet come 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.

Packet Filtering Rules

- A packet filter applies a set of rules to each incoming IP packet and then forwards or discards the packet.
- The packet filter typically sets up a list of rules which may match fields in the IP or TCP header.
- If there is a match to one of the rules, that rule is able to determine whether to forward or discard the packet.
- If there is no match to any rule then two default actions (forward and discard) will be taken.

TELNET packet filtering

- TELNET is a simple remote terminal access that allows a user to log onto a computer across an Internet.
- TELNET establishes a TCP connection and then passes keystrokes from the user's keyboard directly to the remote computer as if they had been typed on a keyboard attached to the remote machine.
- TELNET also carries output from the remote machine back to the user's screen.
- TELNET client software allows the user to specify a remote machine either by giving its domain name or IP address.

Table: Telnet packet-filtering example

Rule number	Action	Source IP	Source port	Destination IP	Destination port	Protocol
1	Discard	*	23	*	*	TCP
2	Discard	*	*	*	23	TCP

If a packet is passed through the filter and has a source port of 23, it will immediately be discarded. If a packet with a destination port of 23 is passed through this filter, it is discarded only after rule 2 has been applied. All other packets will be discarded.

FTP packet filtering

If the FTP service is to apply the same basic rule as applied to TELNET, the packet filter to allow or block FTP would look like following Table. The FTP service is typically associated with using TCP ports 20 and 21.

Table 11.2 FTP packet-filtering example

Rule number	Action	Source IP	Source port	Destination IP	Destination port	Protocol
1	Allow	192.168.10.0	*	*	21	TCP
2	Block	*	20	192.168.10.0	< 1024	TCP
3	Allow	*	20	192.168.10.0	*	TCP
						ACK = 1

One approach to handling FTP connections is explained with the following rule set. Rule 1 allows any host with the network address 192.168.10.0 to initiate a TCP session on any destination IP address on port 21. Rule 2 blocks any packet originating from any remote address with a source port of 20 and contacting a host with a network address 192.168.10.0 on any port less than 1024. Rule 3 allows any remote address that has a source port of 20 and is contacting any host with a network address of 192.168.10.0 on any port. Once a connection is set up, the ACK flag ($ACK = 1$) of a TCP segment is set to acknowledge segments sent from the other side. If any packet violates rule 2, it will be immediately discarded, and rule 3 will never be executed.

Advantages of packet filter router

- Simple
- Transparent to users
- Very fast

Weakness of packet filter firewalls

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.
- It does not support advanced user authentication schemes.
- They are generally vulnerable to attacks such as layer address spoofing.

Attacks

IP address spoofing – The intruders transmit packets from the outside with a source IP address field containing an address of an internal host.

Countermeasure:

To discard packet with an inside source address if the packet arrives on an external interface.

Source routing attacks – the source station specifies the route that a packet should take as it crosses the internet; i.e., it will bypass the firewall.

Countermeasure:

To discard all packets that uses this option.

Tiny fragment attacks – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

Countermeasure:

To discard all packets where the protocol type is TCP and the IP fragment offset is equal to 1.

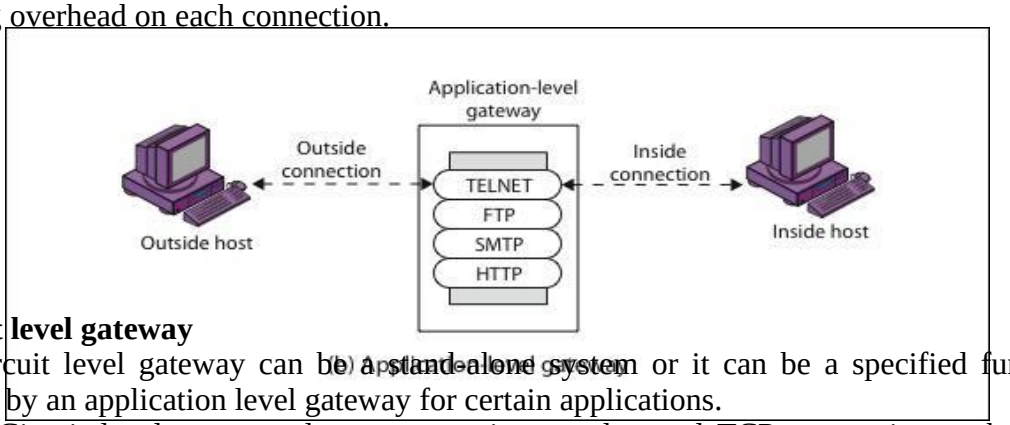
2. Application level gateway

An Application level gateway, also called a proxy server, acts as a relay of application level traffic.

The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed.

When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

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



3. Circuit level gateway

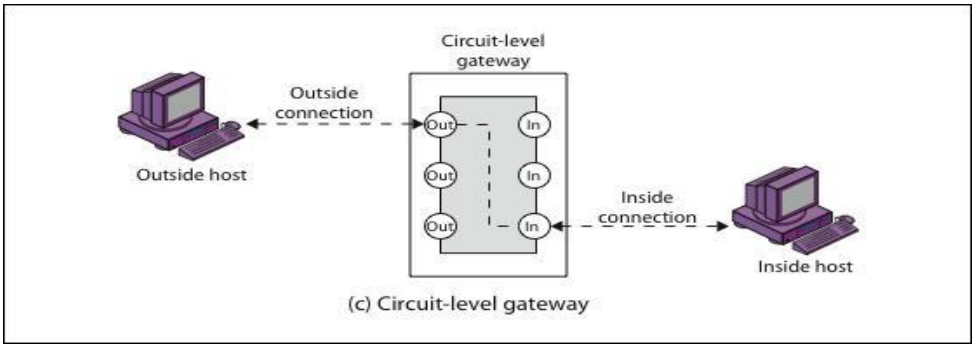
Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications.

A Circuit level gateway does not permit an end-to-end TCP connection; rather, the

gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host.

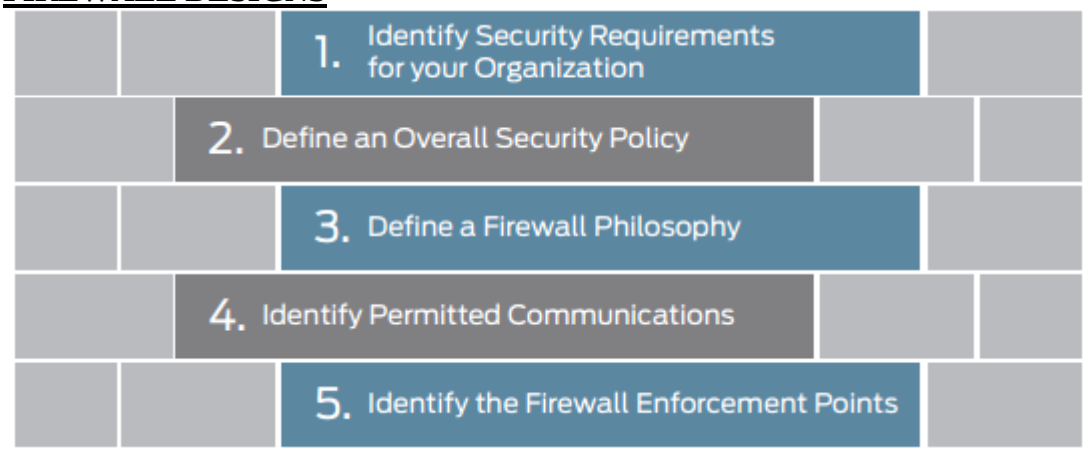
Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.



(c) Circuit-level gateway

FIREWALL DESIGNS



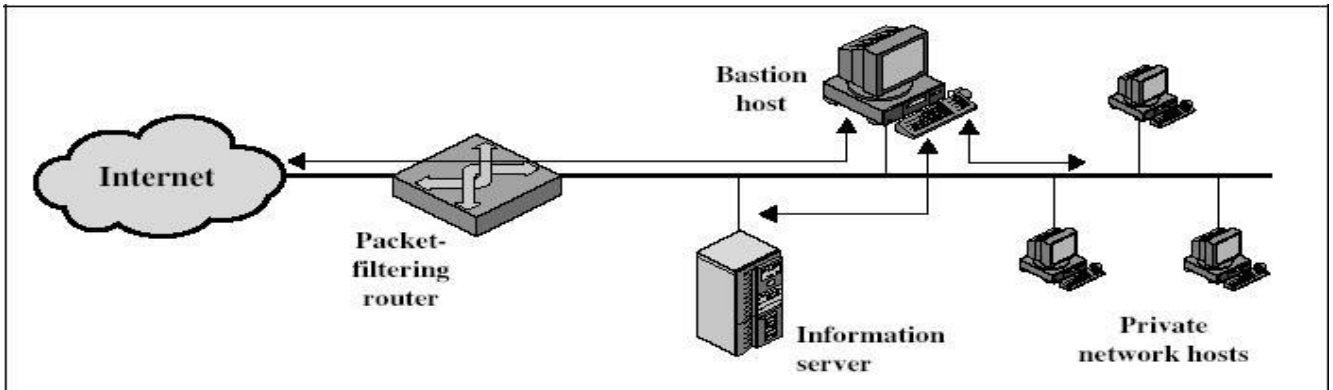
Five best-practice steps to optimal firewall design

The primary step in designing a secure firewall is obviously to prevent the firewall devices from being compromised by threats. To provide a certain level of security, the three basic firewall designs are considered: **a single-homed bastion host, a dual-homed bastion host, and a screened subnet firewall**. The first two options are for creating a screened host firewall, and the third option contains an additional packet-filtering router to achieve another level of security.

1. Screened Host Firewall (Single-Homed Bastion Host)

The first type of firewall is a screened host which uses a single-homed bastion host plus a packet-filtering router.

Single-homed bastion hosts can be configured as either circuit-level or application-level gateways. When using either of these two gateways, each of which is called a *proxy server*, the bastion host can hide the configuration of the internal network.

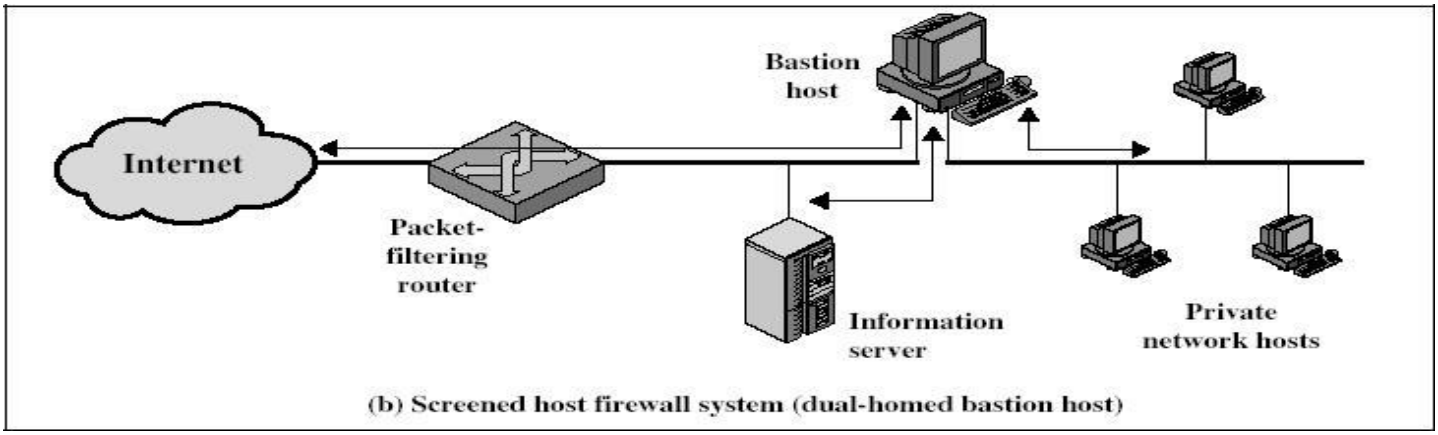


In this configuration, the firewall consists of two systems: a packet filtering router and a bastion host. Typically, the router is configured so that

- For traffic from the internet, only IP packets destined for the bastion host are allowed in.
- 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:
- This configuration implements both packet level and application level filtering, allowing for considerable flexibility in defining security policy.

2. Screened Host Firewall (Dual-Homed Bastion Host)

- The configuration of the screened host firewall using a dual-homed bastion host adds significant security, compared with a single-homed bastion host.
- Dual-homed bastion host has two network interfaces.
- In the previous configuration, if the packet filtering router is compromised, traffic could flow directly through the router between the internet and the other hosts on the private network. This configuration physically prevents such a security break.
- This firewall implementation is secure due to the fact that it creates a complete break between the internal network and the external Internet.



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

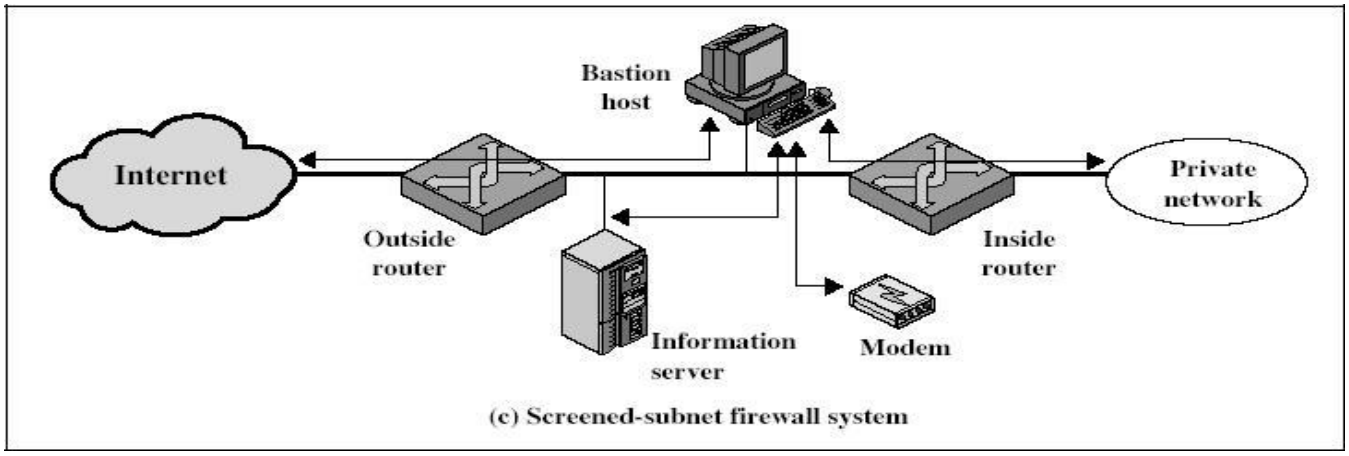
3. Screened Subnet Firewall

- The third implementation of a firewall is the screened subnet, which is also known as a *DMZ.*
- This firewall is the most secure one among the three implementations, simply because it uses a bastion host to support both circuit- and application-level gateways.
- All publicly accessible devices, including modem and server, are placed inside the DMZ.
- The screened subnet firewall contains external and internal screening routers. Each is configured such that its traffic flows only to or from the bastion host.
- This arrangement prevents any traffic from directly traversing the DMZ subnetwork.
- The external screening router uses standard filtering to restrict external access to the bastion host and rejects any traffic that does not come from the bastion host.

- This router also uses filters to prevent attacks such as IP spoofing and source routing.
- The internal screening router also uses rules to prevent spoofing and source routing.

This configuration offers several advantages:

- There are now three levels of defense to thwart intruders.
- The outside router advertises only the existence of the screened subnet to the internet; therefore the internal network is invisible to the internet.
- Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore the systems on the internal network cannot construct direct routes to the internet.



(c) Screened-subnet firewall system

Part –A (2 Marks)

1. What are the services provided by PGP?

- Digital signature
- Message encryption
- Compression
- E-mail compatibility
- Segmentation

2. Explain the reasons for using PGP?

- It is available free worldwide in versions that run on a variety of platforms, including DOS/windows, UNIX, Macintosh and many more.
- It is based on algorithms that have survived extensive public review and are considered extremely secure.
- It has a wide range of applicability from corporations that wish to select and enforce a standardized scheme for encrypting files and communication.
- It was not developed by nor is it controlled by any governmental or standards Organization.

3. Why E-mail compatibility function in PGP needed?

Electronic mail systems only permit the use of blocks consisting of ASCII text.

To accommodate this restriction PGP provides the service converting the row 8- bit binary stream to a stream of printable ASCII characters.

The scheme used for this purpose is Radix-64 conversion.

4. Name any cryptographic keys used in PGP?

- One-time session conventional keys.

- Public keys
- Private keys
- Pass phrase based conventional keys.

5. Define key Identifier?

PGP assigns a key ID to each public key that is very high probability unique with a user ID. It is also required for the PGP digital signature. The key ID associated with each public key consists of its least significant 64bits.

6. List the limitations of SMTP/RFC 822?

- SMTP cannot transmit executable files or binary objects.
- It cannot transmit text data containing national language characters.
- SMTP servers may reject mail message over certain size.
- SMTP gateways cause problems while transmitting ASCII and EBCDIC.
- SMTP gateways to X.400 E-mail network cannot handle non textual data included in X.400 messages.

5. Define S/MIME?

Secure/Multipurpose Internet Mail Extension(S/MIME) is a security enhancement to the MIME Internet E-mail format standard, based on technology from RSA Data Security.

6. What are the elements of MIME?

Five new message header fields are defined which may be included in an RFC 822 header.

A number of content formats are defined.

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

7. What are the headers fields define in MIME?

- MIME version
- Content type
- Content transfer encoding
- Content id
- Content description

8. What are the key algorithms used in S/MIME?

- Digital Signature Standards.
- Diffi Hellman.
- RSA Algorithm.

9. Give the steps for preparing envelope data MIME?

- Generate Ks.
- Encrypt Ks using recipient's public key.
- RSA algorithm used for encryption.
- Prepare the 'recipient info block'.

10. What are the function areas of IP security?[APRIL/MAY 2010]

- Authentication
- Confidentiality
- Key management**.**

11. Give the application of IP security?

- Provide secure communication across private & public LAN.
- Secure remote access over the Internet.
- Secure communication to other organization.

12. Give the benefits of IP security?

- Provide security when IP security implement in router or firewall.
- IP security is below the transport layer is transparent to the application.
- IP security transparent to end-user.
- IP security can provide security for individual user.

13. What are the protocols used to provide IP security?

- Authentication header (AH) protocol.
- Encapsulating Security Payload (ESP).

14. Specify the IP security services?

- Access control
- Connectionless interpretty
- Data origin authentication
- Rejection of replayed packet
- Confidentiality
- Limited traffic for Confidentiality.

15. Give some threats or attacks on web

- Integrity
- Confidentiality
- Denial of service
- Authentication

16. Name some web traffic security approaches

- Networklevel or IPSec approach
- Transport level
- Application level

17. What do you mean by 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.

18. Give some key features of SET

- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

19. List the limitations of SMTP/RFC 822? [Nov'16]

- Transfers limited in size
- Gateways do not always map properly between EBCDIC and ASCII
- Cannot handle non-text data in X.400 messages
- Not all SMTP implementations adhere completely to RFC821 (tabs, truncation, etc)

20. Define Botnets? [Nov'16]

A botnet is a collection of internet-connected devices, which may include PCs, servers, mobile devices and internet of things devices that are infected and controlled by a common type of malware. Users are often unaware of a botnet infecting their system.

21.Define virus. Specify the types of viruses? [NOV2007][MAY 2014]

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

- Parasitic virus
- Memory-resident virus

- Boot sector virus and Stealth virus
- Polymorphic virus

22. What are Honeypots in Intrusion Detection System? [April/May 2010] [Nov/Dec 2013][MAY 2011]

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

- Divert an attacker from accessing critical systems
- Collect information about the attacker's activity
- Encourage the attacker to stay on the system long enough for administrators to respond

23. What is Polymorphic Virus? [Nov/Dec 2013][DEC 2012]

A virus that mutates with every infection, making detection by the "signature" of the virus impossible.

24. What are the two types of audit records? [May/June 2012]

This audit record is a fundamental tool for intrusion detection. Here they record of ongoing activity by users are maintained. This is an input to the intrusion detection system.

- **Native Audit Record:** This is maintained in multiuser operating system.
- **Detection Specific Audit Record:** A facility is implemented to generate the audit record.

25. Define Worm? [MAY 2014]

Program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. In addition to propagation, the worm usually perform some unwanted function

26. What do you mean by Trojan horse?

Trojan horse is a computer virus and is from a link you have clicked and downloaded. Also it can be passed on by e-mail so you have to be very careful whoever sends the mail to you. It will steal your personal information from your computer

27. What is a DDoS? [MAY 2014]

A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack.

A more serious threat is posed by a DDoS attack. In a DDoS attack, an attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target.

28. List the three classes of intruders? [Nov'16]

- Masquerader : pretend to be someone one is not. 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: authentic user doing unauthorized actions. 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 : done secretively, especially because illicit. An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

29. Define Zombie? [Nov'16]

A zombie computer virus is a computer that's been infected by a computer virus or compromised by a hacker. It can be controlled under remote direction to perform criminal tasks, as well as infect other computers with viruses. A zombie computer can appear to be performing normally, making it hard for you to know that your computer has been compromised.

Part-B(16 Marks)

- Write about PGP in detail
- Write about S/MIME in detail
- Explain IP security architecture in detail
- Write about IPSec key management
- Write the applications and benefits of IPSec
- Explain about the threats on web and its consequences and countermeasures
- Write about different web security approaches
- Explain briefly about SSL and TLS
- Discuss the working of SET with neat diagram.[nov'16]
- Discuss the role of compression in the operation of a virus.(16)(MAY 2010)
- How does a worm propagate? Illustrate with an example.(8) (DEC 2010)
- Give the nature and the counter measures followed for the various viruses and related threats.(16)(MAY 2011)
- Write down the four generations of antivirus software. (DEC 2011)
- Draw and discuss about the audit records and metrics used for profiling in intrusion Detection system.(16) (May 2011)(Dec 2011)(MAY 2012)
- Describe in detail about software threats.(16)(Dec 2012)(MAY 2013)
- Write short notes on the following: Types of Viruses.(8) (DEC 2013)
- Explain the technical details of firewall design and describe any two types of firewall with neat diagrams.(16)(Dec 2012)(Dec 2010)(Dec 2011) (DEC 2013) [NOV 2014]
- Explain the approaches used for Intrusion Detection.(8).(May 2011)(Dec 2011)(May 2012)(MAY 2014)(NOV 2014]
- Explain the characteristic of firewall and types of firewall.(11) (Dec 2010)(Dec 2011) (Dec 2013)(Dec 2012) (MAY 2014)[NOV 2014]
- Discuss about various virus countermeasures.(5) (MAY 2014)
- Explain the technical details of firewall and describe any three types of firewall with neat diagram?(Nov'16)
- Explain how secure electronic transaction (SET) protocol enables e-transactions in details. Explain the components involved. (Nov/17)
- Discuss how firewalls help in the establishing a security framework for an organization.(Nov/17)