CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 1 NOTES

UNIT I INTRODUCTION

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

Definition

Cryptography is the science of using mathematics to encrypt and decrypt data.

Phil Zimmermann

Cryptography is the art and science of keeping messages secure.

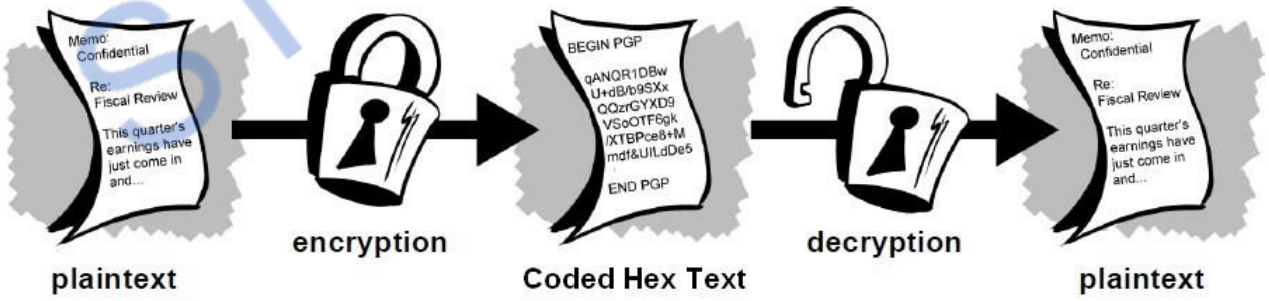
Bruce Schneier

The art and science of concealing the messages to introduce secrecy in information Security is recognized as cryptography.

It is the study and practice of techniques for secure communication in the presence of third parties called adversaries. Data Confidentiality, Data Integrity, Authentication and Non-repudiation are core principles of modern-day cryptography.

Terminologies

A message is **plaintext** (sometimes called clear text). The process of disguising a message in such a way as to hide its substance is **encryption**. An encrypted message is **cipher text**. The process of turning cipher text back into plaintext is **decryption**.



A **cryptosystem** is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a **cipher system**. The various components of a basic cryptosystem are as follows

- Plaintext
- Encryption Algorithm
- Cipher text
- Decryption Algorithm
- Encryption Key
- Decryption Key

While **cryptography** is the science of securing data, **cryptanalysis** is the science of analyzing and breaking secure communication. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. **Cryptanalysts** are also called attackers. **Cryptology** embraces both cryptography and cryptanalysis.

Security Trends

Definition of Computer Security

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

Confidentiality

- **Data confidentiality**
  Assures that private or confidential information is not made available or disclosed to unauthorized
- **Privacy**
  Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Integrity

- **Data integrity**
  Assures that information and programs are changed only in a specified and authorized manner.
- **System integrity**
  Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Availability

- Assures that systems work promptly and service is not denied to authorize users.

CIA Triad

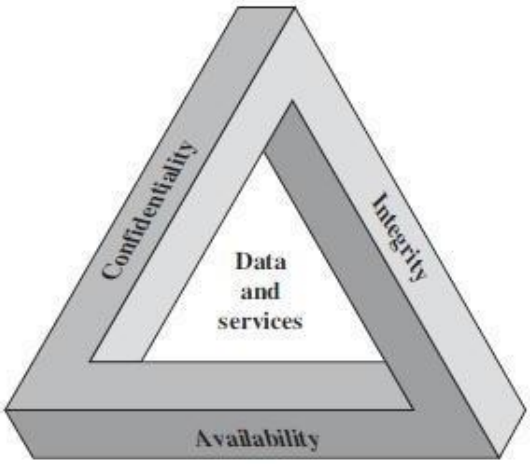


Figure 1.1 The Security Requirements Triad

Confidentiality

- Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information.
- A loss of confidentiality is the unauthorized disclosure of information.

Integrity

- Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity.
- A loss of integrity is the unauthorized modification or destruction of information.

Availability
Ensuring timely and reliable access to and use of information

A loss of availability is the disruption of access to or use of information or an information system.

Authenticity

- The property of being genuine and being able to be verified and trusted

Accountability

- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity

Legal, Ethical and Professional aspects of security

We must understand the scope of an organization's legal and ethical responsibilities. To minimize liabilities/reduce risks, the security practitioner must:

1. Understand current legal environment.
2. Stay current with laws and regulations.
3. Watch for new issues and emerge.

Information is endangered both by external factors, such as hackers, computer viruses, thefts, and internal ones - the loss of data as a result of improper protection, the lack of backup copies or the loss of a flash drive that contains unprotected data. An improper protection of data may result in the loss of company's reputation, its customers' trust or in financial losses. This issue is of particular importance as regards the court system due to the volume of personal data that are processed and stored in courts and their unique character (sentences, orders, and statements of reasons, convictions, and personal details of victims or land registers). They all constitute information that must be protected against theft, loss or alterations. The loss of data could affect negatively the trial and the judicial independence by possible external pressure in cases where data was lost.

Law and Ethics in Security

- Definition of Laws
- Rules that mandate or prohibit certain behavior
- Drawn from ethics
- Definition of Ethics
- Define socially acceptable behaviors
- Key difference between law and ethics
- Laws carry the authority of a governing body
- Ethics do not carry the authority of a governing body
- Based on cultural mores
- Fixed moral attitudes or customs
- Some ethics standards are universal

Organizational Liability and the Need for Counsel

What if an organization does not behave ethically? Even if there is no breach of criminal law, there can still be liability.

- What is Liability?
- Legal obligation of organization
- Extends beyond criminal or contract law
- Include legal obligation to restitution
- Employee acting with or without the authorization performs and illegal or unethical act that causes some degree of harm
- Employer can be held financially liable

An organization increases its liability if it refuses to take measures known as due care.

- What is Due care?
- Organization makes sure that every employee knows what is acceptable or unacceptable
- Knows the consequences of illegal or unethical actions
- Due diligence
- Requires
- Make a valid effort to protect others
- Maintains the effort

In legal system, any court can assert its authority over an individual or organization if it can establish jurisdiction

- What is Jurisdiction?
- Court's right to hear a case if a wrong is committed
- Term – long arm
- Extends across the country or around the world

Policy Versus law

- Policies
- Guidelines that describe acceptable and unacceptable employee behaviors
- Functions as organizational laws
- Has penalties, judicial practices, and sanctions
- Difference between policy and law
- Ignorance of policy is acceptable
- Ignorance of law is unacceptable
- Keys for a policy to be enforceable
- Dissemination (distribution)
- Review (reading)
- Comprehension (understanding)
- Compliance (agreement)
- Uniform enforcement

Types of Law

- Civil – govern a nation or state
- Criminal – addresses activities and conduct harmful to public
- Private – encompasses family, commercial, labor, and regulates the relationship between individuals and organizations
- Public – regulates the structure and administration of government agencies and their relationships with citizens, employees, and other governments

International Laws and Legal Bodies

- Organizations do business on the Internet – they do business globally
- Professionals must be sensitive to the laws and ethical values of many different cultures, societies, and countries
- Few international laws relating to privacy and informational security
- International laws are limited in their enforceability

Ethics and Security

The Ten Commandments of Computer Ethics[6]

From The Computer Ethics Institute

1. Thou shalt not use a computer to harm other people.
2. Thou shalt not interfere with other people's computer work.
3. Thou shalt not snoop around in other people's computer files.
4. Thou shalt not use a computer to steal.
5. Thou shalt not use a computer to bear false witness.
6. Thou shalt not copy or use proprietary software for which you have not paid.
7. Thou shalt not use other people's computer resources without authorization or proper compensation.
8. Thou shalt not appropriate other people's intellectual output.
9. Thou shalt think about the social consequences of the program you are writing or the system you are designing.
10. Thou shalt always use a computer in ways that ensure consideration and respect for your fellow humans.

Overriding factor in leveling ethical perceptions within a small population is education

- Employees must be trained in expected behaviors of an ethical employee, especially in areas of information security
- Proper ethical training vital to creating informed, well prepared, and low-risk system user

Deterrence To Unethical And Illegal Behaviours

- Deterrence: best method for preventing an illegal or unethical activity; e.g., laws, policies, technical controls
- Laws and policies only deter if three conditions are present:
    - Fear of penalty
    - Probability of being caught
    - Probability of penalty being administered

Codes of Ethics And Professional Organizations

- Several professional organizations have established codes of conduct/ethics
- Codes of ethics can have positive effect unfortunately, many employers do not encourage joining of these professional organizations
- Responsibility of security professionals to act ethically and according to policies of employer, professional organization, and laws of society

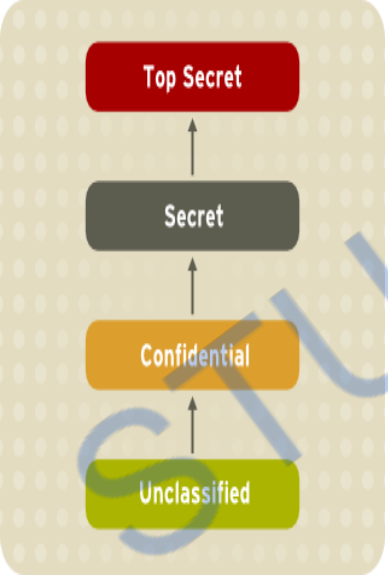
Need for Multilevel Security

Having information of different security levels on the same computer systems poses a real threat. It is not a straight-forward matter to isolate different information security levels, even though different users log in using different accounts, with different permissions and different access controls.

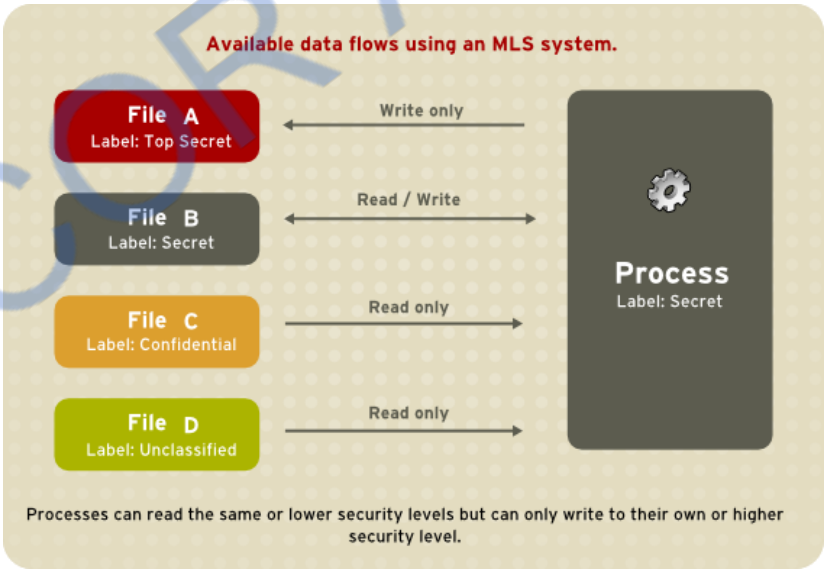
Some organizations go as far as to purchase dedicated systems for each security level. This is often prohibitively expensive, however. A mechanism is required to enable users at different security levels to access systems simultaneously, without fear of information contamination.

The term multi-level arises from the defense community's security classifications: Confidential, Secret, and Top Secret.

Individuals must be granted appropriate clearances before they can see classified information. Those with Confidential clearance are only authorized to view Confidential documents; they are not trusted to look at Secret or Top Secret information. The rules that apply to data flow operate from lower levels to higher levels, and never the reverse. This is illustrated below.



Information Security Levels



Available data flow using MLS system

Under such a system, users, computers, and networks use labels to indicate security levels. Data can flow between like levels, for example between "Secret" and "Secret", or from a lower level to a higher level. This means that users at level "Secret" can share data with one another, and can also retrieve information from Confidential-level (i.e., lower-level), users. However, data cannot flow from a higher level to a lower level. This prevents processes at the "Secret" level from viewing information classified as "Top Secret". It also prevents processes at

a higher level from accidentally writing information to a lower level. This is referred to as the "no read up, no write down" model.

MLS and System Privilege

MLS access rules are always combined with conventional access permissions (file permissions). For example, if a user with a security level of "Secret" uses Discretionary Access Control (DAC) to block access to a file by other users, this also blocks access by users with a security level of "Top Secret". A higher security clearance does not automatically give permission to arbitrarily browse a file system.

Users with top-level clearances do not automatically acquire administrative rights on multi-level systems. While they may have access to all information on the computer, this is different from having administrative rights.

Security Levels, Objects and Subjects

As discussed above, subjects and objects are labeled with Security Levels (SLs), which are composed of two types of entities:

Sensitivity: — A hierarchical attribute such as "Secret" or "Top Secret".

Categories: — A set of non-hierarchical attributes such as "US Only" or "UFO".

An SL must have one sensitivity, and may have zero or more categories.

Examples of SLs are: { Secret / UFO, Crypto }, { Top Secret / UFO, Crypto, Stargate } and { Unclassified }

Note the hierarchical sensitivity followed by zero or more categories. The reason for having categories as well as sensitivities is so that sensitivities can be further compartmentalized on a need-to-know basis.

Security Polices

Following are some points which help in security policy of an organization.

- Who should have access to the system?
- How it should be configured?
- How to communicate with third parties or systems?

Policies are divided in two categories −

- User policies
- IT policies.

User policies generally define the limit of the users towards the computer resources in a workplace. For example, what are they allowed to install in their computer, if they can use removable storages.

Whereas, IT policies are designed for IT department, to secure the procedures and functions of IT fields.

- **General Policies** − This is the policy which defines the rights of the staff and access level to the systems. Generally, it is included even in the communication protocol as a preventive measure in case there are any disasters.
- **Server Policies** − This defines who should have access to the specific server and with what rights. Which software's should be installed, level of access to internet, how they should be updated.
- **Firewall Access and Configuration Policies** − It defines who should have access to the firewall and what type of access, like monitoring, rules change. Which ports and services should be allowed and if it should be inbound or outbound.
- **Backup Policies** − It defines who is the responsible person for backup, what should be the backup, where it should be backed up, how long it should be kept and the frequency of the backup.
- **VPN Policies** − These policies generally go with the firewall policy, it defines those users who should have a VPN access and with what rights. For site-to-site connections with partners, it defines the access level of the partner to your network, type of encryption to be set.

Structure of a Security Policy

When you compile a security policy you should have in mind a basic structure in order to make something practical. Some of the main points which have to be taken into consideration are −

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.

Types of Policies

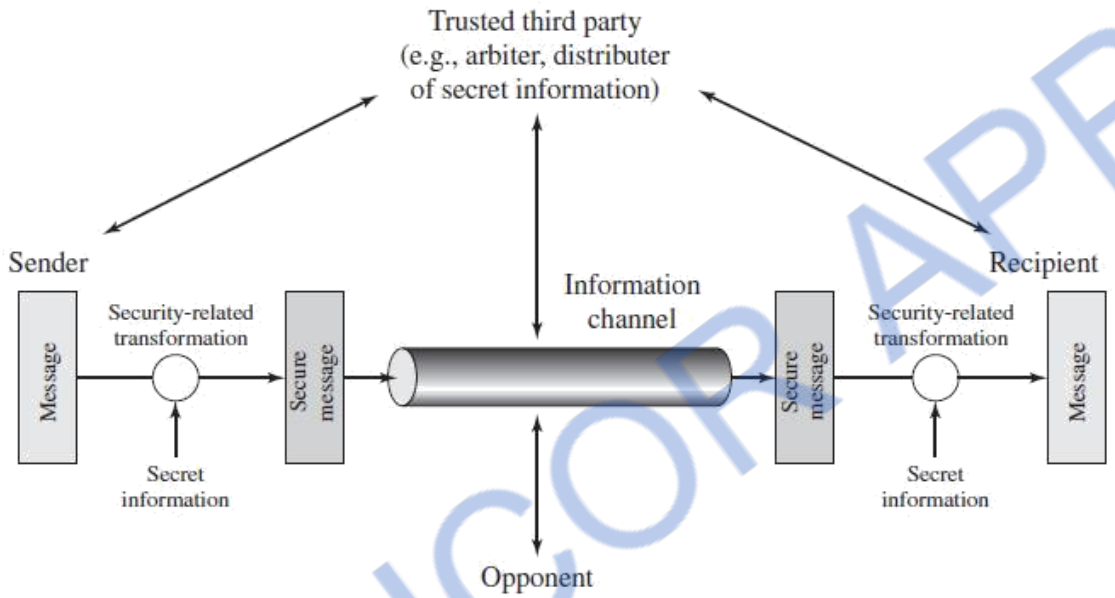
In this section we will see the most important types of policies.

- **Permissive Policy** − It is a medium restriction policy where we as an administrator block just some well-known ports of malware regarding internet access and just some exploits are taken in consideration.

- **Prudent Policy** − This is a high restriction policy where everything is blocked regarding the internet access, just a small list of websites are allowed, and now extra services are allowed in computers to be installed and logs are maintained for every user.
- **Acceptance User Policy** − This policy regulates the behavior of the users towards a system or network or even a webpage, so it is explicitly said what a user can do and cannot in a system. Like are they allowed to share access codes, can they share resources, etc.
- **User Account Policy** − This policy defines what a user should do in order to have or maintain another user in a specific system. For example, accessing an e-commerce webpage. To create this policy, you should answer some questions such as −
    - Should the password be complex or not?
    - What age should the users have?
    - Maximum allowed tries or fails to log in?
    - When the user should be deleted, activated, blocked?
- **Information Protection Policy** − This policy is to regulate access to information, hot to process information, how to store and how it should be transferred.
- **Remote Access Policy** − This policy is mainly for big companies where the user and their branches are outside their headquarters. It tells what should the users access, when they can work and on which software like SSH, VPN, RDP.
- **Firewall Management Policy** − This policy has explicitly to do with its management, which ports should be blocked, what updates should be taken, how to make changes in the firewall, how long should be the logs be kept.
- **Special Access Policy** − This policy is intended to keep people under control and monitor the special privileges in their systems and the purpose as to why they have it. These employees can be team leaders, managers, senior managers, system administrators, and such high designation based people.
- **Network Policy** − This policy is to restrict the access of anyone towards the network resource and make clear who all will access the network. It will also ensure whether that person should be authenticated or not. This policy also includes other aspects like, who will authorize the new devices that will be connected with network? The documentation of network changes. Web filters and the levels of access. Who should have wireless connection and the type of authentication, validity of connection session?
- **Email Usage Policy** − This is one of the most important policies that should be done because many users use the work email for personal purposes as well. As a result information can leak outside. Some of the key points of this policy are the employees should know the importance of this system that they have the privilege to use. They should not open any attachments that look suspicious. Private and confidential data should not be sent via any encrypted email.
- **Software Security Policy** − This policy has to do with the software's installed in the user computer and what they should have. Some of the key points of this policy are Software of the company should not be given to third parties. Only the white list of software's should

be allowed, no other software's should be installed in the computer. Warez and pirated software's should not be allowed

Model for Network Security



A message is to be transferred from one party to another across some sort of Internet service.
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: encryption of the message, addition of a code based on the contents

Some secret information shared by the two principals, unknown to the opponent
Example: encryption key used in conjunction with the transformation

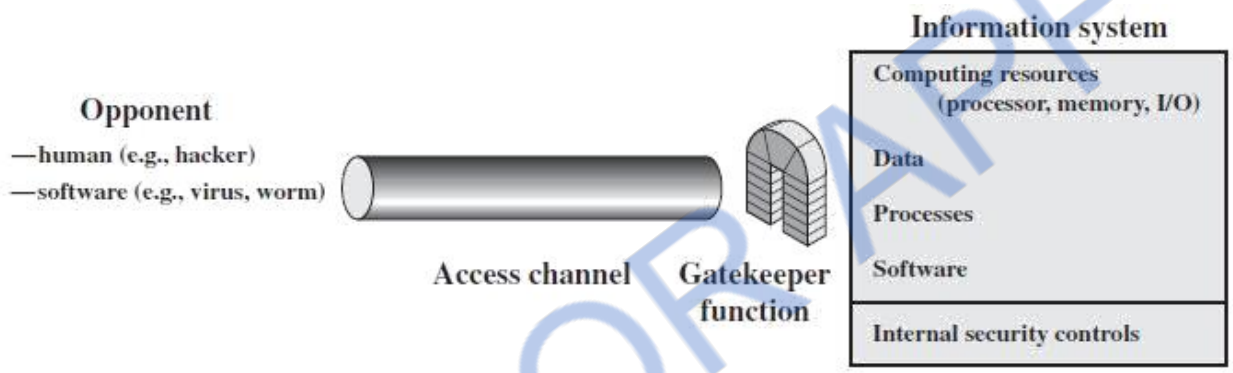
A trusted third party may be needed to achieve secure transmission.

- for distributing the secret information to the two principals
- to arbitrate disputes between the two principals concerning the authenticity of a message transmission

Four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation 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

Network Access Security Model



- Protecting an information system from unwanted access from hacker, intruder hacker who, with no malign intent, simply gets satisfaction from breaking and entering a computer system.
- Intruder can be a disgruntled employee who wishes to do damage or a Criminal who seeks to exploit computer assets for financial gain
- placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers

Two kinds of threats:

- **Information access threats**: Intercept or modify data on behalf of users who should not have access
- **Service threats**: Exploit service flaws in computers to inhibit use by legitimate users

Examples: Viruses and worms, spread using disks & inserted over network

The OSI Security Architecture

- ITU-T Recommendation X.800, Security Architecture for OSI, defines such a systematic approach
- 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) that is designed to detect, prevent, or recover from a security attack.

Security service

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

Security Attacks

- means of classifying security attacks, used both in X.800 and RFC 2828
- A passive attack attempts to learn or make use of information but does not affect system resources.
- An active attack attempts to alter system resources or affect their operation.

Passive Attacks

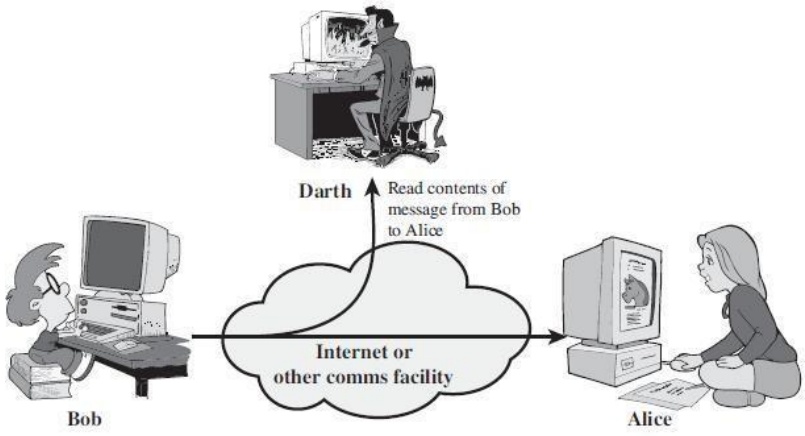
- It is the nature of eavesdropping on, or monitoring of, transmissions.
- The goal is to obtain information that is being transmitted.
- very difficult to detect, because they do not involve any alteration of the data
- feasible to prevent the success of these attacks, usually by means of encryption
- emphasis in dealing with passive attacks is on prevention rather than detection

Two types of passive attacks

- Release of message contents
- Traffic analysis.

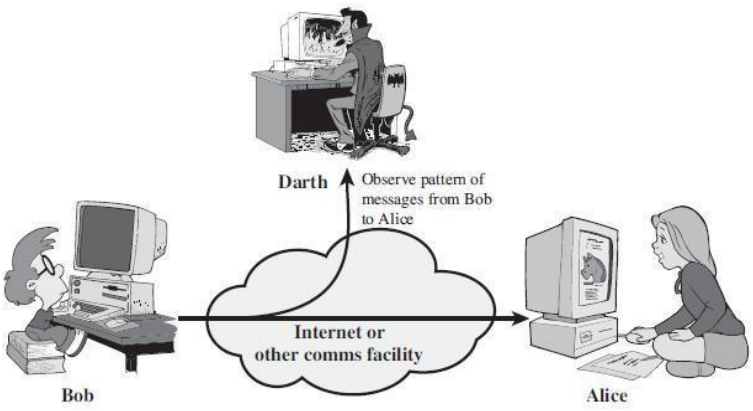
Release of Message Contents

- A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or



confidential information

- prevent an opponent from learning the contents of these transmissions



Traffic Analysis

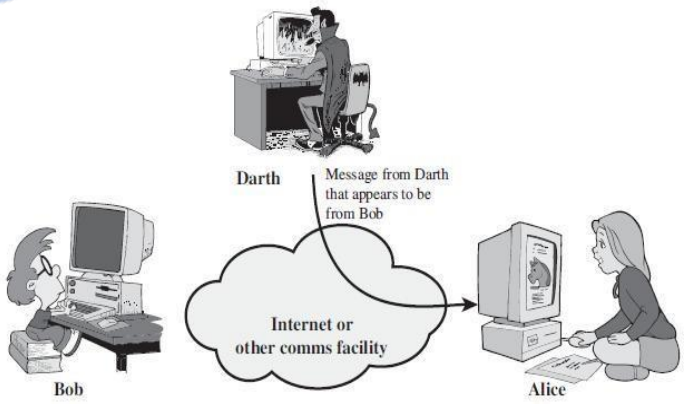
- observe the pattern of these messages
- The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.
- This information might be useful in guessing the nature of the communication that was taking place

Active Attacks

- Active attacks involve some modification of the data stream or the creation of a false stream
- detect and to recover from any disruption or delays caused by them
- can be subdivided into four categories:
  - masquerade,
  - replay,
  - modification of messages
  - denial of service

Masquerade

- one entity pretends to be a different entity
- usually includes one of the other forms of active attack

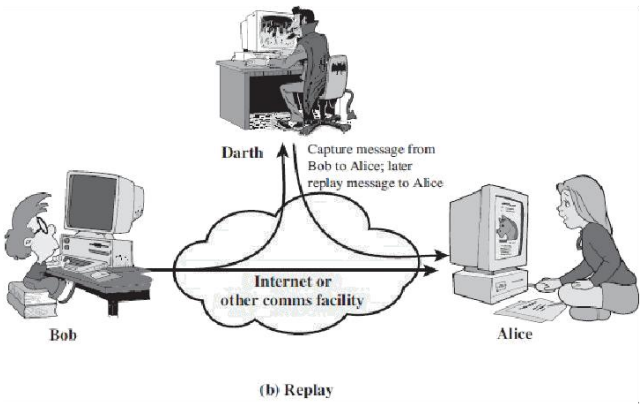


Example

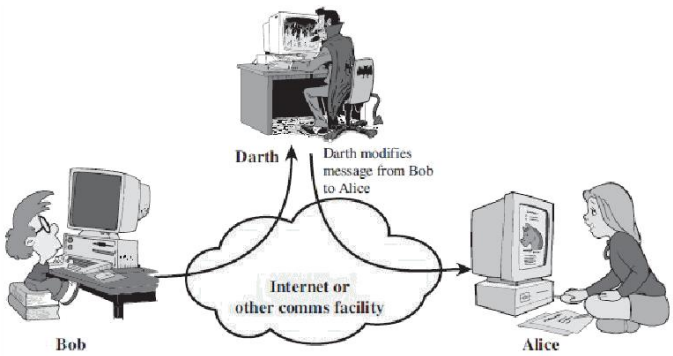
Authentication sequences can be captured and replayed after a valid authentication sequence

Replay

- passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect



(b) Replay



(c) Modification of messages

Modification of Messages

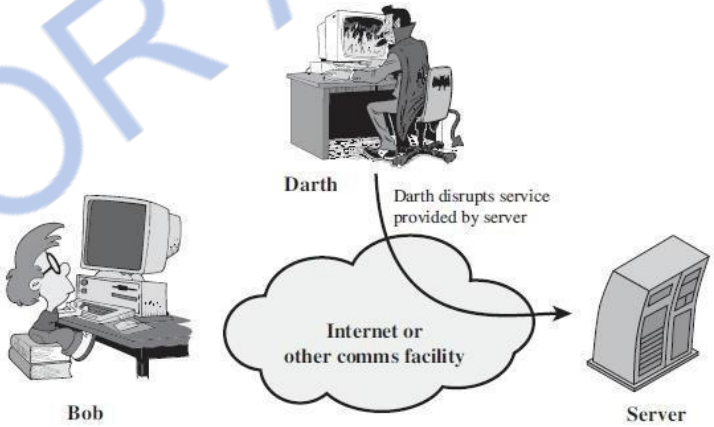
- Some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect

Example

- A message meaning "Allow John Smith to read confidential file accounts" is modified to mean "Allow Fred Brown to read confidential file accounts."

Denial of Service

- Prevents or inhibits the normal use or management of communications facilities
- May have a specific target; for example, an entity may suppress all messages directed to a particular destination
- Disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance



(d) Denial of service

Security Services in X.800

- X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers.
- RFC 2828, defines as a processing or communication service that is provided by a system to give a specific kind of protection to system resources;
- Security services implement security policies and are implemented by security mechanisms.

X.800

- divides these services into five categories and fourteen specific services

Authentication

- The assurance that the communicating entity is the one that it claims to be
- Two types
  - Peer Entity Authentication
  - Data-Origin Authentication

Access control

- The prevention of unauthorized use of a resource

Data confidentiality

- The protection of data from unauthorized disclosure.
- Four Types
  - Connection Confidentiality
  - Connectionless Confidentiality
  - Selective-Field Confidentiality
  - Traffic-Flow Confidentiality

Data integrity

- The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
- Five types
  - Connection Integrity with Recovery
  - Connection Integrity without Recovery
  - Selective-Field Connection Integrity
  - Connectionless Integrity
  - Selective-Field Connectionless Integrity

Nonrepudiation

- Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication
- Two types
  - Nonrepudiation, Origin
  - Nonrepudiation, Destination

Security Mechanisms in X.800.

Specific security mechanisms:

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

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.

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).

Access Control
A variety of mechanisms that enforce access rights to resources.

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

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

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

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

Notarization
The use of a trusted third party to assure certain properties of a data exchange.

Pervasive Security Mechanisms
Mechanisms that are not specific to any particular OSI security service or protocol layer.

Trusted Functionality
That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

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

Event Detection
Detection of security-relevant events.

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

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

Classical Encryption Techniques

- Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption.
- Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements.
- Rotor machines are sophisticated precomputer hardware devices that use substitution techniques.
- Steganography is a technique for hiding a secret message within a larger one in such a way that others cannot discern the presence or contents of the hidden message.

- Symmetric Cipher Model

o Cryptanalysis and Brute-Force Attack

- Substitution Techniques
  - Caesar Cipher

  - Monoalphabetic Ciphers
  - Playfair Cipher
  - Hill Cipher
  - Polyalphabetic Ciphers
  - One-Time Pad
- Transposition Techniques
- Rotor Machines
- Steganography

Introduction

- **Symmetric encryption** is a form of cryptosystem in which encryption and decryption are performed using the **same key**. It is also known as **conventional encryption**.
- Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- The two types of attack on an encryption algorithm are **cryptanalysis**,based on properties of the encryption algorithm, and **brute-force**, which involves trying all possible keys.
- Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements.
- Rotor machines are sophisticated precomputer hardware devices that use substitution techniques.
- Steganography is a technique for hiding a secret message within a larger one in such a way that others cannot discern the presence or contents of the hidden message.
- An original message is known as the **plaintext**, while the coded message is called the **ciphertext**.
- The process of converting from plaintext to ciphertext is known as **enciphering or encryption**; restoring the plaintext from the ciphertext is **deciphering or decryption**.
- 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**.
- 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

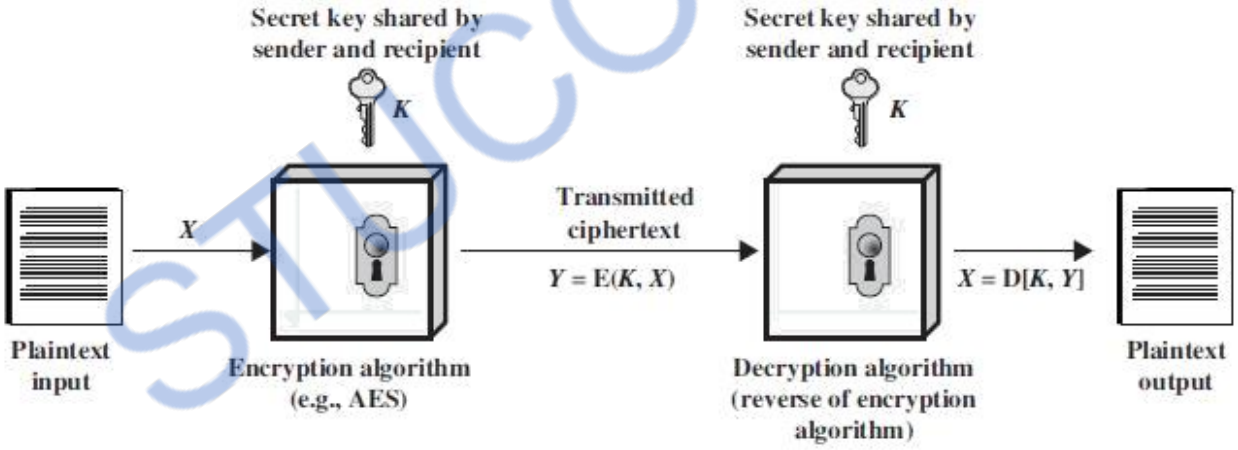
"breaking the code."The areas of cryptography and cryptanalysis together are called **cryptology**

Symmetric Cipher Model

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.
- **Ciphertext:** 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

Simplified Model of Symmetric Encryption



Two requirements for secure use of conventional / symmetric encryption

- We need a strong encryption algorithm

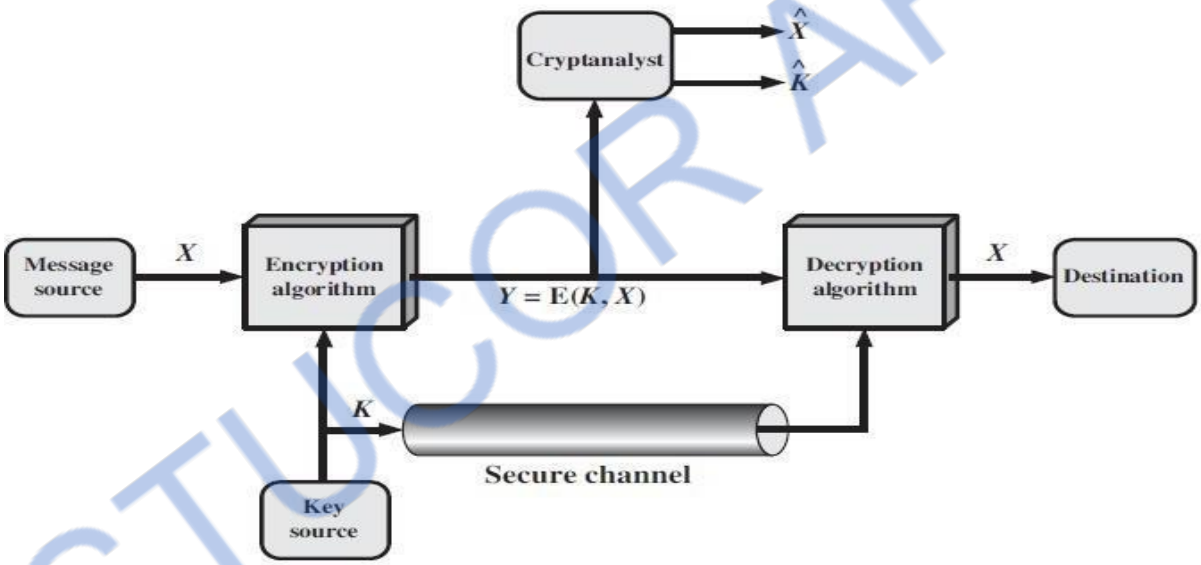
  The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext

- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all

communication using this key is readable, do not need to keep the algorithm secret; we need to keep only the key secret. The principal security problem is maintaining the secrecy of the key

Model of Conventional Cryptosystem

A source produces a message in plaintext, X = [X1, X2, ..., XM]. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet {0, 1} is typically used. For encryption, a key of the form K = [K1, K2, ..., KJ] 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 ciphertext Y = [Y1, Y2, ..., YN]. We can write this as

Y = E(K, X)

This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.

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

X = D(K, Y)

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D)

algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate.

Substitution Techniques

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

1. Caesar Cipher

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

plain: meet me after the toga party

cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A.

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

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

For each plaintext P substitute the ciphertext letter C

C = E(3, p) = (p + 3) mod 26

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

C = E(k, p) = (p + k) mod 26

where takes on a value in the range 1 to 25.The decryption algorithm is simply

p = D(k, C) = (C - k) mod 26

Cryptanalysis of Caesar Cipher

- only have 26 possible ciphers
- A maps to A,B,..Z
- could simply try each in turn
- a brute force search
- given ciphertext, just try all shifts of letters
- do need to recognize when have plaintext

Exercise:

Plain Text : civil engineering

Key : 7

Cipher Text :

Cipher Text : RTXYFRFENSLUJWXTSGFPMJNX

Key : 5

Plain Text : ?

Monoalphabetic Ciphers

- Rather than just shifting the alphabet shuffle (jumble) the letters arbitrarily
- Each plaintext letter maps to a different random ciphertext letter
- Hence key is 26 letters long
- The “cipher” line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4x1026 possible keys.
- This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute- force techniques for cryptanalysis
- Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet
- A countermeasure is to provide multiple substitutes, known as homophones, for a single letter.
- For example, the letter e could be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone assigned to a letter in rotation or randomly

Language Redundancy and Cryptanalysis

- human languages are redundant
- eg "th lrd s m shphrd shll nt wnt"
- letters are not equally commonly used
- in English E is by far the most common letter
- followed by T,R,N,I,O,A,S
- other letters like Z,J,K,Q,X are fairly rare
- have tables of single, double & triple letter frequencies for various languages
- two-letter combinations, known as digrams (ex: th)

2. Playfair Cipher

The best-known multiple-letter encryption cipher is the Playfair. The Playfair algorithm is based on the use of a 5×5 matrix of letters constructed using a keyword.

Playfair Key Matrix

- 5 × 5 matrix of letters constructed using a keyword
- filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom,
- filling in the remainder matrix with the remaining letters in alphabetic order.
- The letters I and J count as one letter
- Example matrix using the keyword MONARCHY

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

In this case, the keyword is monarchy. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter. Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

2. 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.

3. 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.

4. 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 / JM.

Exercise:

Plain Text : civil engineering

Key : abishek

Cipher Text : ?

Note: The letters M and N count as one letter

Example

Given the key MONARCHY apply Play fair cipher to plain text "FACTIONALISM"

Solution

(p) FA CT IO NA LI SM

(c) IO DL FA AR SE LA

(d) FA CT IO NA LI SM

Security of Playfair Cipher

- security much improved over monoalphabetic since have 26 x 26 = 676 digrams
- would need a 676 entry frequency table to analyse and correspondingly more ciphertext
- was widely used for many years eg. by US & British military in WW1

it can be broken, given a few hundred letters since still has much of plaintext structure

3. Hill Cipher

Another interesting multiletter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929.

This encryption algorithm takes successive M plaintext letters and substitutes for them M ciphertext letters. The substitution is determined by linear equations in which each character is assigned a numerical value (a=0, b=1, c=2,....., z=25). For M=3, the system can be described as

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

$$(c_1\ c_2\ c_3) = (p\ p_2\ p_3)\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

C = **PK** mod 26

where **C** and **P** are row vectors of length 3 representing the plaintext and ciphertext, and **K** is a 3 * 3 matrix representing the encryption key. Operations are performed mod 26.

Example:

Plain Text : paymoremoney

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The first three letters of the Plain Text are represented by

$$\begin{vmatrix} 15 \\ 0 \\ 24 \end{vmatrix} \text{ then, } K \begin{vmatrix} 15 \\ 0 \\ 24 \end{vmatrix} = \begin{vmatrix} 375 \\ 879 \\ 486 \end{vmatrix} \text{mod26} = \begin{vmatrix} 11 \\ 13 \\ 18 \end{vmatrix} = \text{LNS}$$

Cipher Text : LNSHDLEWMTRW

Exercise:

Plain Text : FINALYEAR

$$\begin{vmatrix} 2 & 5 & 3 \\ 3 & 1 & 4 \\ 9 & 7 & 6 \end{vmatrix}$$ Key :

Cipher Text : ?

Cipher Text : XAJOCVDAIUSGDAAUPIAGDGCSGDHAFQGSXI

Key : abishek

Plain Text : ?

Note: The letters M and N count as one letter

Example

Encrypt the message “meet me at the usual place at ten rather than eight oclock” using the Hill cipher with the key ( ). Show your calculations and the result.Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext.

1) mathematically give each letter a number

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
0	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

2) 1st pair from plain text "me" => $\begin{pmatrix}12\\4\end{pmatrix}$

$$\begin{pmatrix}9 & 4\\5 & 7\end{pmatrix}\begin{pmatrix}12\\4\end{pmatrix} => \begin{pmatrix}9x12+4x4\\5x12+7x4\end{pmatrix} = \begin{pmatrix}124\\88\end{pmatrix} => mod\ 26 => \begin{pmatrix}20\\10\end{pmatrix} => \begin{pmatrix}u\\k\end{pmatrix}$$

3) 2nd pair fro plain text "et"

$$\begin{pmatrix}9 & 4\\5 & 7\end{pmatrix}\begin{pmatrix}4\\19\end{pmatrix} => \begin{pmatrix}9x4+4x19\\5x4+7x19\end{pmatrix} = \begin{pmatrix}112\\153\end{pmatrix} => mod\ 26 => \begin{pmatrix}8\\23\end{pmatrix} => \begin{pmatrix}i\\x\end{pmatrix}$$

4) Cipher text for "meet" is "ukix"

5) To get plain text from cipher text, we need to find the inverse of K

6) |A| = (9x7 – 5x4) => 43

7) Adj (A) => $\begin{pmatrix}7 & -4\\-5 & 9\end{pmatrix} => \frac{1}{43}\begin{pmatrix}7 & -4\\-5 & 9\end{pmatrix} => \frac{1}{17}\begin{pmatrix}7 & -4\\-5 & 9\end{pmatrix}$ ($\because 43\ \%\ 26 = 17$)

8) Find the multiplier for 17, using 17 x X = 1 mod 26 => X = 23

9) $\begin{pmatrix}161 & -92\\-115 & 207\end{pmatrix} => mod\ 26 => \begin{pmatrix}5 & -14\\-11 & 25\end{pmatrix} => \begin{pmatrix}\mathbf{5} & \mathbf{12}\\\mathbf{15} & \mathbf{25}\end{pmatrix}$ ($\because Add\ 26\ for - ive\ values$)

10) P = CK^{-1} = > For the ciper text of “uk”,

$$\begin{pmatrix}5 & 12\\15 & 25\end{pmatrix}\begin{pmatrix}20\\10\end{pmatrix} = \begin{pmatrix}5x20+12x10\\15x20+25x10\end{pmatrix} => \begin{pmatrix}220\\550\end{pmatrix} mod\ 26 => \begin{pmatrix}12\\4\end{pmatrix} = \begin{pmatrix}m\\e\end{pmatrix}$$

Hence the plain text is “me”

4. Polyalphabetic Ciphers

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**.

All these techniques have the following features in common:

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

5. Vigenere Cipher

Encryption and Decryption

Given a key letter X and plaintext letter Y, the ciphertext letter is at the intersection of the row labeled X and the column labled Y.

To encrypt a message, a key is needed that is as long as the message. Usually the key is repeating keyword. Decryption is simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is the top of the column.

Example:

Key : deceptive
Plain Text : we are discovered yourself

key: deceptivedeceptivedeceptive
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Exercise:

Plaintext : cryptography and network security
Key : sectionb
Ciphertext : ?

	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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

Vigenere Table

Note : Rows represents Plaintext and Columns represents the Key

keyword can be eliminated by using a nonrepeating keyword that is as long as the message itself.Vigenère proposed what is referred to as an **autokey system**, in which a keyword is concatenated with the plaintext itself to provide a running key. For our example,

key: deceptivewearediscoveredsav
plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

Exercise:
Plaintext : cryptography and network security
Key : sectionb
Ciphertext : ?

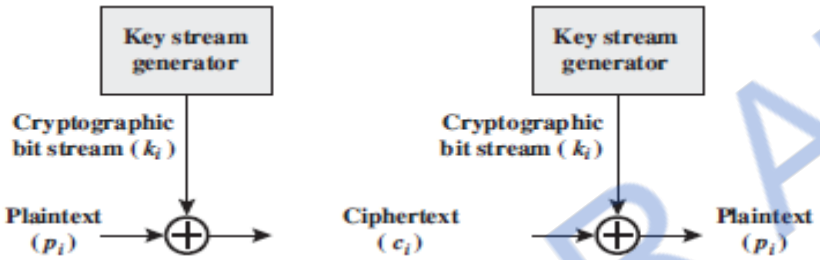
Vernam Cipher

The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918.

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation



6. One-Time Pad

- improvement to the Vernam cipher that yields the ultimate in security
- using a random key that is as long as the message, so that the key need not be repeated
- the key is to be used to encrypt and decrypt a single message, and then is discarded.
- Each new message requires a new key of the same length as the new message **Example** ciphertext:ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS key: pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih plaintext: mr mustard with the candlestick in the hall ciphertext:ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS key: mfugpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt plaintext: miss scarlet with the knife in the library **two fundamental difficulties**
- problem of making large quantities of random keys
- problem of key distribution and protection

Transposition Techniques

A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters

Rail Fence Technique

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

For example, to encipher the message "meet me after the toga party" with a rail fence of depth 2, we write the following

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

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

Pure Transposition Cipher

Write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns.

The order of the columns then becomes the key to the algorithm

Example

Key: 4 3 1 2 5 6 7

Plaintext: a t t a c k p

o s t p o n e d

u n t i l t w o

a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Double Transposition

performing more than one stage of transposition Example

if the foregoing message is reencrypted using the same algorithm

Key: 4 3 1 2 5 6 7

Input: t t n a a p t

m t s u o a o

d w c o i x k

n l y p e t z

Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

This is a much less structured permutation and is much more difficult to cryptanalyze

Steganography

We conclude with a discussion of a technique that is, strictly speaking, not encryption, namely, steganography

A plaintext message may be hidden in one of two ways.

- The methods of steganography conceal the existence of the message
- The methods of cryptography render the message unintelligible to outsiders o by various transformations of the text

Various ways to conceal the message

Arrangement of words or letters within an apparently innocuous text spells out the real message

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

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

Hiding a message by using the least significant bits of frames on a CD

- The Kodak Photo CD format's maximum resolution is 2048 by 3072 pixels, with each pixel containing 24 bits of RGB color information.

- The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image
- Thus you can hide a 2.3-megabyte message in a single digital snapshot

Number of drawbacks

- lot of overhead to hide a relatively few bits of information
- once the system is discovered, it becomes virtually worthless
- the insertion method depends on some sort of key

o Alternatively, a message can be first encrypted and then hidden using steganography

Advantage of steganography

- can be employed by parties who have something to lose should the fact of their secret communication be discovered
- Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide

Foundations of 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.

Characteristics of Modern Cryptography

There are three major characteristics that separate modern cryptography from the classical approach.

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'. The techniques employed for coding were kept secret and only the parties involved in communication knew about them.	It relies on publicly known mathematical algorithms for coding the information. Secrecy is obtained through a secrete key which is used as the seed for the algorithms. The computational difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information even if he knows the algorithm used for coding.
It requires the entire cryptosystem for communicating confidentially.	Modern cryptography requires parties interested in secure communication to possess the secret key only.

Perfect Security

What is a “secure” cipher? Intuitively, the answer is that a secure cipher is one for which an encrypted message remains “well hidden,” even after seeing its encryption.

However, turning this intuitive answer into one that is both mathematically meaningful and practically relevant is a real challenge. Indeed, although ciphers have been used for centuries, it is only in the last few decades that mathematically acceptable definitions of security have been developed.

In this section, we develop the mathematical notion of perfect security — this is the “gold standard” for security (at least, when we are only worried about encrypting a single message and do not care about integrity). We will also see that it is possible to achieve this level of security; indeed, we will show that the one-time pad satisfies the definition. However, the one-time pad is not very practical, in the sense that the keys must be as long as the messages: if Alice wants to send a 1GB file to Bob, they must already share a 1GB key! Unfortunately, this cannot be avoided: we will also prove that any perfectly secure cipher must have a key space at least as large as its message space. This fact provides the motivation for developing a definition of security that is weaker, but that is acceptable from a practical point of view, and which allows one to encrypt long messages using short keys.

If Alice encrypts a message m under a key k, and an eavesdropping adversary obtains the cipher text c, Alice only has a hope of keeping m secret if the key k is hard to guess, and that means, at the very least, that the key k should be chosen at random from a large key space. To say that m is “well hidden” must at least mean that it is hard to completely determine m from c, without knowledge of k; however, this is not really enough. Even though the adversary may not know k, we assume that he does know the encryption algorithm and the distribution of k. In fact, we will assume that when a message is encrypted, the key k is always chosen at random, uniformly from among all keys in the key space. The adversary may also have some knowledge of the message encrypted — because of circumstances, he may know that the set of possible messages is quite small, and he may know something about how likely each possible message is. For example, suppose he knows the message m is either m0 = "ATTACK AT DAWN" or m1 = "ATTACK AT DUSK", and that based on the adversary’s available intelligence, Alice is equally likely to choose either one of these two messages. This, without seeing the cipher text c, the adversary would only have a 50% chance of guessing which message Alice sent. But we are assuming the adversary does know c. Even with this knowledge, both messages may be possible; that is, there may exist keys k0 and k1 such that E(k0,m0) = c and E(k1,m1) = c, so he cannot be sure if m = m0 or m = m1.

However, he can still guess. Perhaps it is a property of the cipher that there are 800 keys k0 such that E(k0,m0) = c, and 600 keys k1 such that E(k1,m1) = c. If that is the case, the adversary’s best guess would be that m = m0. Indeed, the probability that this guess is correct is equal to 800/(800 + 600) ⇡ 57%, which is better than the 50% chance he would have without knowledge of the ciphertext. Our formal definition of perfect security expressly rules out the possibility that knowledge of the ciphertext increases the probability of guessing the encrypted message, or for that matter, determining any property of the message whatsoever.

Without further ado, we formally define perfect security. In this definition, we will consider a probabilistic experiment in which is key is drawn uniformly from the key space. We write k to denote the random variable representing this random key. For a message m, E(k,m) is another random variable, which represents the application of the encryption function to our random key and the message m. Thus, every message m gives rise to a different random variable E(k,m).

Definition of Perfect Security

Let ε= (E,D) be a Shannon cipher defined over (K,M, C).
Consider a probabilistic experiment in which the random variable k is uniformly distributed over K. If for all m0,m1 Є M, and all c Є C, we have

Pr [E(k,m0) = c] = Pr [E(k,m1) = c], then we say that ε is a perfectly secure Shannon cipher.

There are a number of equivalent formulations of perfect security

Theorem 2.4. *Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Consider a random experiment in which* $\mathbf{k}$ *and* $\mathbf{m}$ *are random variables, such that*

- $\mathbf{k}$ *is uniformly distributed over* $\mathcal{K}$,
- $\mathbf{m}$ *is distributed over* $\mathcal{M}$, *and*
- $\mathbf{k}$ *and* $\mathbf{m}$ *are independent.*

Define the random variable $\mathbf{c} := E(\mathbf{k}, \mathbf{m})$. *Then we have:*

- *if* $\mathcal{E}$ *is perfectly secure, then* $\mathbf{c}$ *and* $\mathbf{m}$ *are independent;*
- *conversely, if* $\mathbf{c}$ *and* $\mathbf{m}$ *are independent, and each message in* $\mathcal{M}$ *occurs with nonzero probability, then* $\mathcal{E}$ *is perfectly secure.*

Proof. We define $\mathcal{M}^*$ to be the set of messages that occur with nonzero probability.

We begin with a simple observation. Consider any fixed $m \in \mathcal{M}^*$ and $c \in \mathcal{C}$. Then we have

$$\Pr[\mathbf{c} = c \mid \mathbf{m} = m] = \Pr[E(\mathbf{k}, m) = c \mid \mathbf{m} = m],$$

and since $\mathbf{k}$ and $\mathbf{m}$ are independent, so are $E(\mathbf{k}, m)$ and $\mathbf{m}$, and hence

$$\Pr[E(\mathbf{k}, m) = c \mid \mathbf{m} = m] = \Pr[E(\mathbf{k}, m) = c].$$

Putting this all together, we have:

$$\Pr[\mathbf{c} = c \mid \mathbf{m} = m] = \Pr[E(\mathbf{k}, m) = c]. \tag{2.1}$$

We now prove the first implication. So assume that $\mathcal{E}$ is perfectly secure. We want to show that $\mathbf{c}$ and $\mathbf{m}$ are independent. To to this, let $m \in \mathcal{M}^*$ and $c \in \mathcal{C}$ be given. It will suffice to show that

$$\Pr[\mathbf{c} = c \mid \mathbf{m} = m] = \Pr[\mathbf{c} = c].$$

We have

$$\begin{aligned}
\Pr[\mathbf{c} = c] &= \sum_{m' \in \mathcal{M}^*} \Pr[\mathbf{c} = c \mid \mathbf{m} = m'] \Pr[\mathbf{m} = m'] && \text{(by total probability)} \\
&= \sum_{m' \in \mathcal{M}^*} \Pr[E(\mathbf{k}, m') = c] \Pr[\mathbf{m} = m'] && \text{(by (2.1))} \\
&= \sum_{m' \in \mathcal{M}} \Pr[E(\mathbf{k}, m) = c] \Pr[\mathbf{m} = m'] && \text{(by the definition of perfect security)} \\
&= \Pr[E(\mathbf{k}, m) = c] \sum_{m' \in \mathcal{M}^*} \Pr[\mathbf{m} = m'] \\
&= \Pr[E(\mathbf{k}, m) = c] && \text{(probabilities sum to 1)} \\
&= \Pr[\mathbf{c} = c \mid \mathbf{m} = m] && \text{(again by (2.1))}
\end{aligned}$$

This shows that $\mathbf{c}$ and $\mathbf{m}$ are independent.

That proves the first implication. For the second, we assume that $\mathbf{c}$ and $\mathbf{m}$ are independent, and moreover, that every message occurs with nonzero probability (so $\mathcal{M}^* = \mathcal{M}$). We want to show that $\mathcal{E}$ is perfectly secure, which means that for each $m_0, m_1 \in \mathcal{M}$, and each $c \in \mathcal{C}$, we have

$$\Pr[E(\mathbf{k}, m_0) = c] = \Pr[E(\mathbf{k}, m_1) = c]. \tag{2.2}$$

But we have

$$\begin{aligned}
\Pr[E(\mathbf{k}, m_0) = c] &= \Pr[\mathbf{c} = c \mid \mathbf{m} = m_0] \qquad \text{(by (2.1))} \\
&= \Pr[\mathbf{c} = c] \qquad \text{(by independence of } \mathbf{c} \text{ and } \mathbf{m}) \\
&= \Pr[\mathbf{c} = c \mid \mathbf{m} = m_1] \qquad \text{(again by independence of } \mathbf{c} \text{ and } \mathbf{m}) \\
&= \Pr[E(\mathbf{k}, m_1) = c] \qquad \text{(again by (2.1))}.
\end{aligned}$$

That shows that $\mathcal{E}$ is perfectly secure. □

Information Theory

For a long time, information theory has mainly been used in cryptography to prove lower bounds on the size of the secret key required to achieve a certain level of security in secrecy and authentication systems. In order to prove the security of a cryptographic system, a definition of security or, alternatively, of breaking the system must be given. Whether a system with provable security is satisfactory from a theoretical and practical viewpoint depends in a crucial manner on three aspects:

(1) On the acceptability and generality of the definition of security;

(2) On how realistic the two assumptions are; and

(3) On the practicality of the system.

For instance, it is trivial to \prove" the security of a cipher if we define security (irrelevantly) to mean that an adversary is unable to square a circle with straightedge and compass. It is similarly trivial to prove that an adversary cannot obtain any information about the plaintext for a system in which the legitimate receiver cannot either, or if one assumes that the adversary is unable to even receive the ciphertext.

There are two possible types of assumptions about the adversary's computing power:

- A system is called computationally-secure if it is secure against an adversary with reasonably bounded computational resources and it is called information-theoretically secure if it is secure even against adversaries with infinite computing power
- The second type of assumption, namely that an adversary has infinite computing power, implies no restriction whatsoever and therefore anticipates all arguments about model's of computation and realistic estimates of an opponent's computing power.

However, if one considers the theoretical possibility of testing all possible keys of a system at once, it appears impossible to prove a system secure under such an assumption. Here is where information theory comes into play. Shannon defined a cipher system to be perfect if the ciphertext provides no information about the plaintext or, equivalently, if plaintext and ciphertext are

statistically independent. In other words, when a perfect cipher is used to encipher a message, an adversary can do no better than guess the message without even looking at the ciphertext.

The role of information theory in cryptography can be characterized as that of deriving results on the provable security of a system, even in presence of adversaries with infinite computing power.

Basics of Information Theory

According to Shannon, the entropy of an information source S is defined as:

$$H(S) = \eta = \sum_i \; p_i \log_2 \frac{1}{p_i}$$

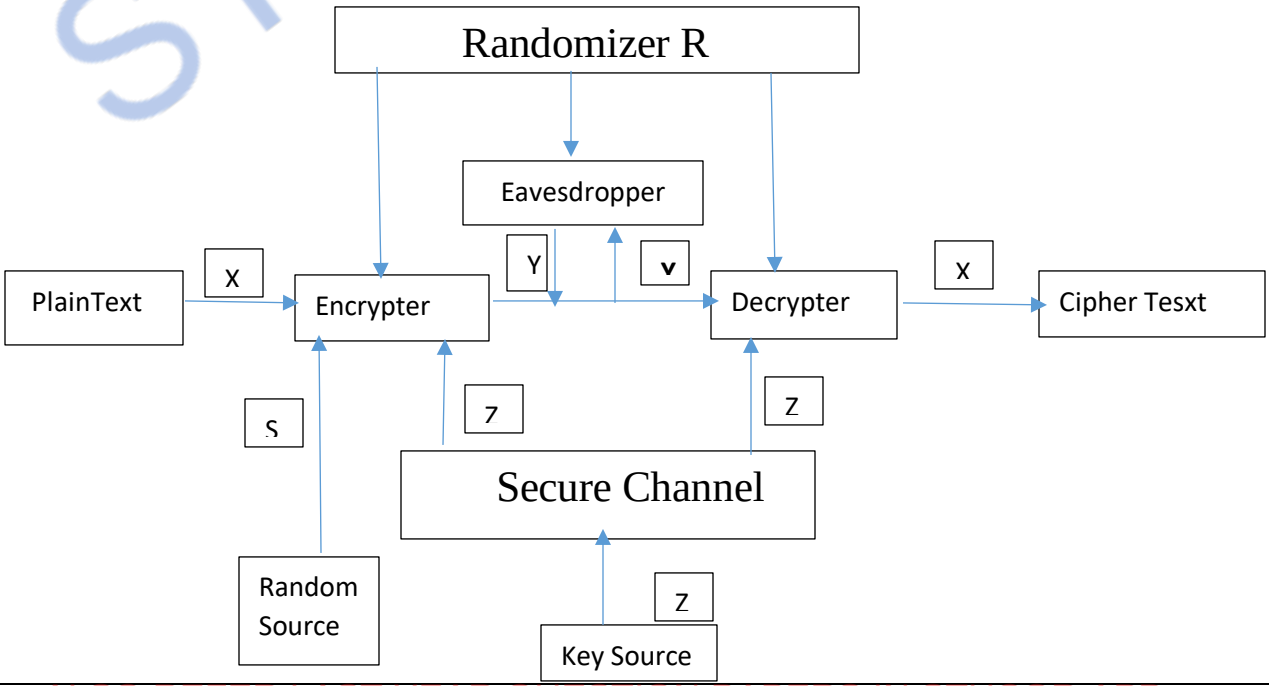
where pi is the probability that symbol Si in S will occur.

- $\log_2 \frac{1}{p_i}$ indicates the amount of information contained in Si, i.e., the number of bits needed to code Si.
- For example, in an image with uniform distribution of gray-level intensity, i.e. pi = 1/256, then the number of bits needed to code each gray level is 8 bits. The entropy of this image is 8.

Pessimistics Results: Lower bounds on key size

Information theory has been used in cryptography primarily to derive pessimistic results, i.e., lower bounds on the size of the secret key necessary to achieve a certain level of security. In this section we review the three most important areas for which such bounds have been derived: secrecy, Y This is a generalization of Shannon's model: it contains a secret randomizer S known only to the sender of a message X as well as a public randomizer R assumed to be available to everybody, including the eavesdropper.

Model of symmetric cipher with two types of Randomizer



There are two dual and complementary security goals in communication: Confidentiality (or secrecy) and authenticity. Confidentiality means that an eavesdropper cannot obtain any useful information about the plaintext, and authenticity means that 5 an active eavesdropper cannot successfully insert a fraudulent message **Y** that will be accepted by the receiver

Information-theoretic security

Information-theoretic security is a cryptosystem whose security derives purely from information theory; the system cannot be broken even if the adversary has unlimited computing power. The cryptosystem is considered cryptanalytically unbreakable if the adversary does not have enough information to break the encryption.

There are a variety of cryptographic tasks for which information-theoretic security is a meaningful and useful requirement. A few of these are:

1. Secret sharing schemes such as Shamir's are information-theoretically secure (and also perfectly secure) in that having less than the requisite number of shares of the secret provides no information about the secret.
2. More generally, secure multiparty computation protocols often have information-theoretic security.
3. Private information retrieval with multiple databases can be achieved with information-theoretic privacy for the user's query.
4. Symmetric encryption can be constructed under an information-theoretic notion of security called entropic security, which assumes that the adversary knows almost nothing about the message being sent. The goal here is to hide all functions of the plaintext rather than all information about it.
5. Quantum cryptography is largely part of information-theoretic cryptography.

Product Cryptosystem

Two of the first kinds of cryptosystems that we considered were simple substitution ciphers and permutation ciphers. Each of them quickly proved vulnerable to attack. We now consider a new kind of cryptosystem that is based on them but which is considerably more difficult to attack; so difficult, in fact, that most modern cryptosystems are of the type we now consider. A product cryptosystem is a block cipher that repeatedly performs substitutions and permutations, one after the other, to produce ciphertext.

Example : DES and AES (Brief description in Unit II)

Cryptanalysis and Brute-Force Attack

Cryptanalysis

Cryptanalysis is the science of cracking codes and decoding secrets. It is used to violate authentication schemes, to break cryptographic protocols, and, more benignly, to find and correct weaknesses in encryption algorithms.

It may be used in information warfare applications - for example, forging an encrypted signal to be accepted as authentic. Competitors who have been able to discover the key will now want to use it to their advantage, therefore they will want to send bogus encrypted messages to the source in order to gain information or gain an advantage. It could also be used to pretend to be the source in order to send bogus information to others, who now will think that it came from the official source.

According to Diffie and Hellman
Skill in the production of cryptanalysis has always been heavily on the side of the professionals, but innovation, particularly in the design of new types of cryptographic systems, has come primarily from amateurs.

Among the types of attacks are:

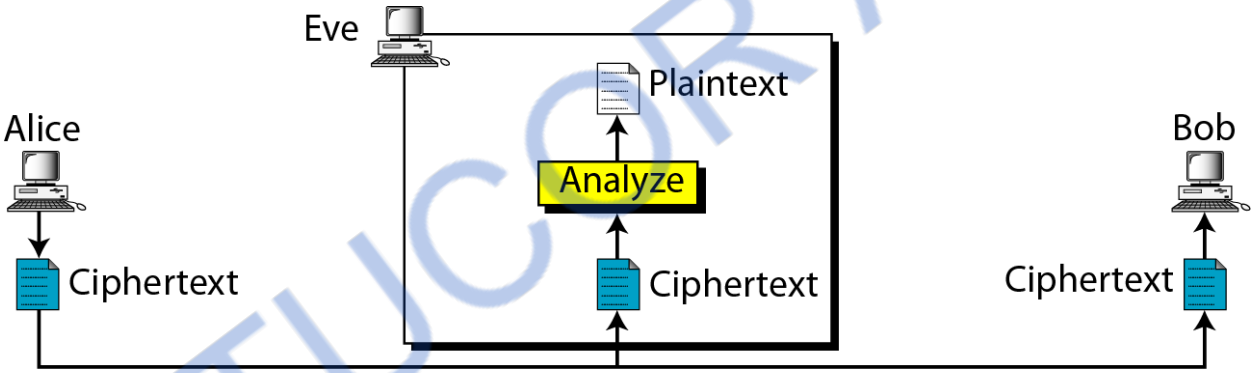
Ciphertext only attacks
Known plaintext attacks
Chosen plaintext attacks
Chosen ciphertext attacks
Man-in-the-middle attacks
Side channel attacks
Brute force attacks
Birthday attacks

Type of Attack	**Known to Cryptanalyst**
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertex	• Encryption algorithm

	• Ciphertext • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Ciphertext Only

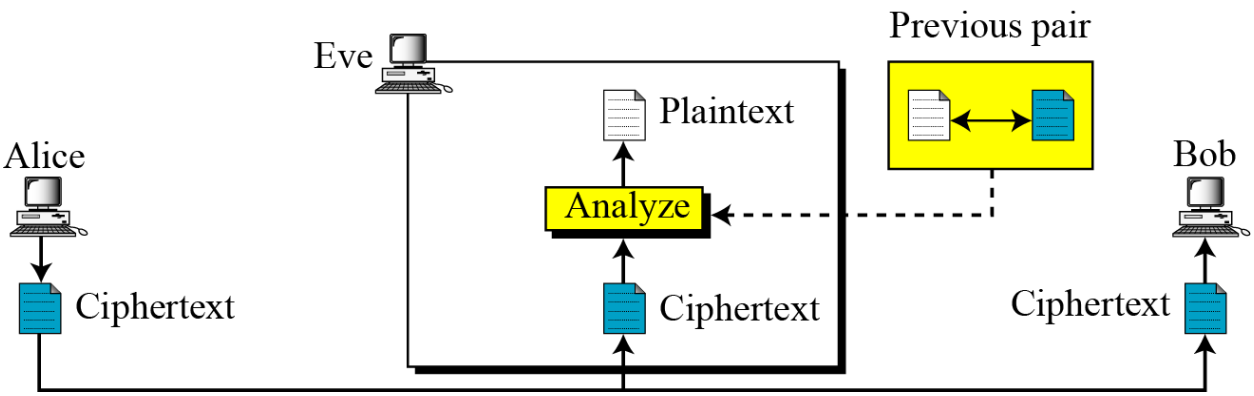
A ciphertext only attack (COA) is a case in which only the encrypted message is available for attack, but because the language is known a frequency analysis could be attempted. In this situation the attacker does not know anything about the contents of the message, and must work from ciphertext only.



Known Plaintext Attack

In a known plaintext attack (KPA) both the plaintext and matching ciphertext are available for use in discovering the key.

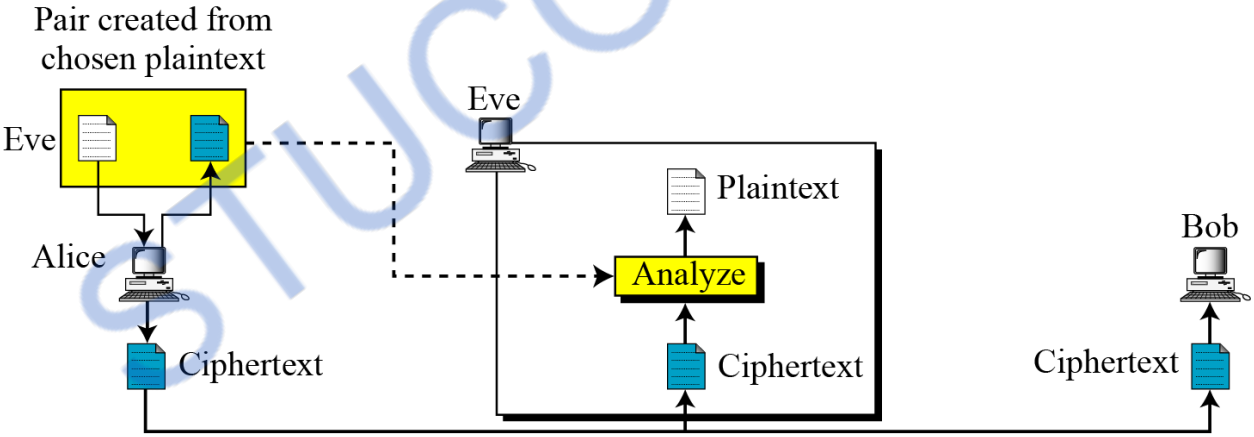
The attacker knows or can guess the plaintext for some parts of the ciphertext. For example, maybe all secure login sessions begin with the characters LOGIN, and the next transmission may be PASSWORD. The task is to decrypt the rest of the ciphertext blocks using this information. This may be done by determining the key used to encrypt the data, or via some shortcut.



Chosen Plaintext Attack

A chosen plaintext attack (CPA) occurs when the attacker gains access to the target encryption device - if, for example, it is left unattended. The attacker then runs various pieces of plaintext though the device for encryption. This is compared to the plaintext to attempt to derive the key.

In an adaptive chosen plaintext attack (ACPA), the attacker not only has access to the plaintext and its encryption, but can adapt or modify the chosen plaintext as needed based on results of the previous encryptions.

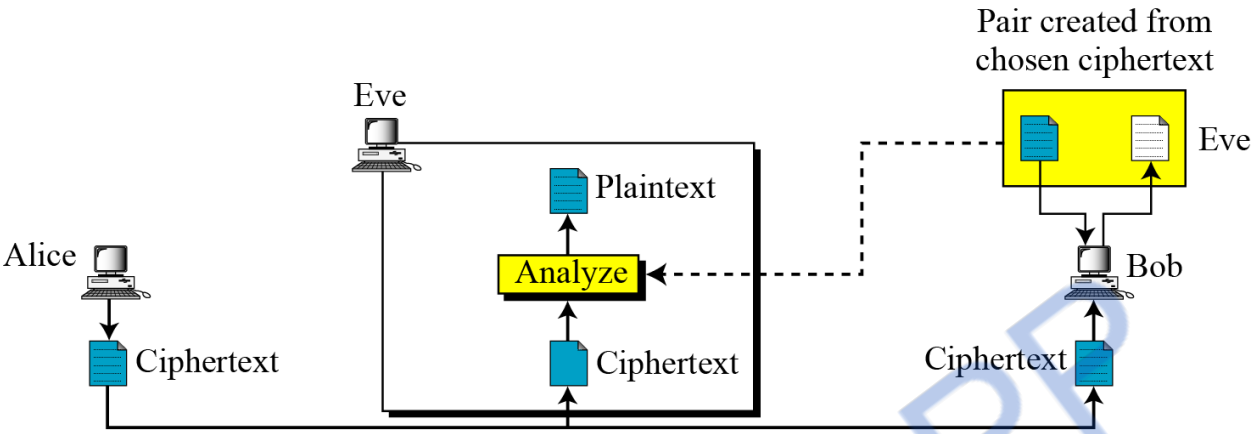


Chosen Ciphertext Attack

In a chosen ciphertext attack (CCA), the cryptanalyst can choose different ciphertexts to be decrypted and has access to the decrypted plaintext.

This type of attack is generally applicable to attacks against public key cryptosystems.

An adaptive chosen ciphertext attack involves the attacker selecting certain ciphertexts to be decrypted, then using the results of these decryptions to select subsequent ciphertexts. The modifications in the ciphertext help in deciphering the key from the decryptions.



Two schemes

- **Unconditionally secure**

If the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available

- **Computationally secure**

meets either of the following criteria:

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

Brute-force attack

- The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.
- On average, half of all possible keys must be tried to achieve success.

CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

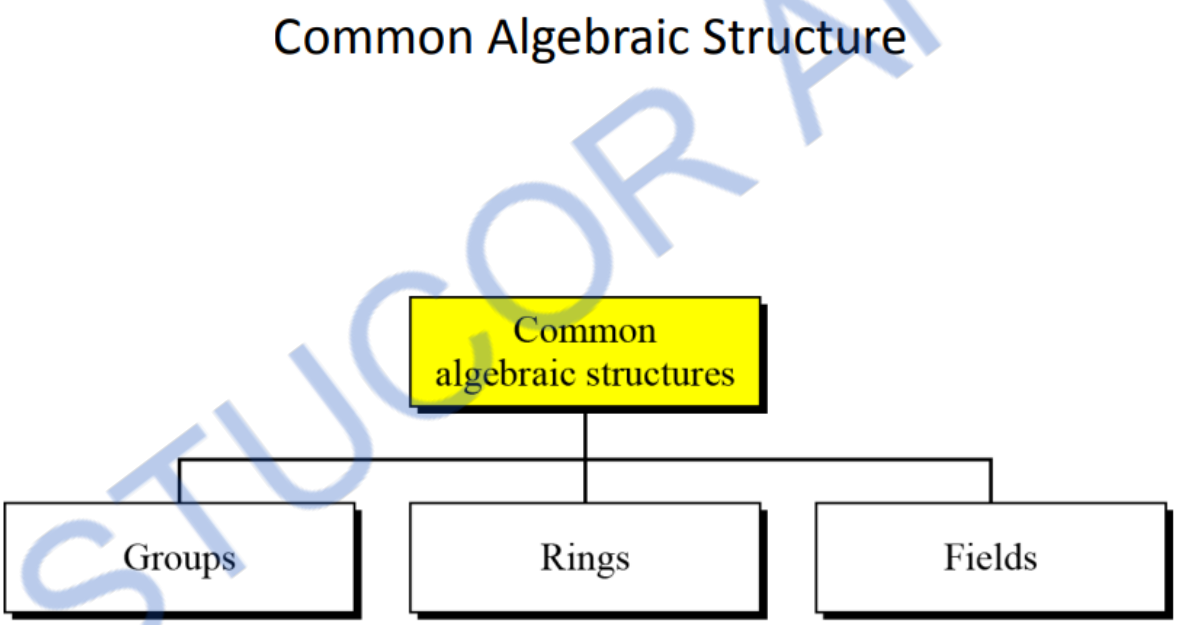
UNIT 2 NOTES

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.

2.1 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. In this topic, we will define three common algebraic structures: groups, rings, and fields.



Modular Arithmetic

The Modulus

If a is an integer and n is a positive integer, we define $a \bmod n$ to be the remainder when a is divided by n. The integer n is called the **modulus**. Thus, for any integer a, we can rewrite Equation (4.1) as follows:

$$a = qn + r \qquad 0 \leq r < n;\ q = \lfloor a/n \rfloor$$

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

Two integers a and b are said to be **congruent modulo** n, if $(a \bmod n) = (b \bmod n)$. This is written as $a \equiv b \pmod{n}$.[2]

$$73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}$$

Note that if $a \equiv 0 \pmod{n}$, then $n|a$.

Properties of Congruences

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

To demonstrate the first point, if $n|(a - b)$, then $(a - b) = kn$ for some k. So we can write $a = b + kn$. Therefore, $(a \bmod n) =$ (remainder when $b + kn$ is divided by n) = (remainder when b is divided by n) $= (b \bmod n)$.

$23 \equiv 8 \pmod{5}$ because $23 - 8 = 15 = 5 \times 3$
$-11 \equiv 5 \pmod{8}$ because $-11 - 5 = -16 = 8 \times (-2)$
$81 \equiv 0 \pmod{27}$ because $81 - 0 = 81 = 27 \times 3$

Modular Arithmetic Operations

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

We demonstrate the first property. Define $(a \bmod n) = r_a$ and $(b \bmod n) = r_b$. Then we can write $a = r_a + jn$ for some integer j and $b = r_b + kn$ for some integer k. Then

$$\begin{aligned}(a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k + j)n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

Examples

$11 \bmod 8 = 3; 15 \bmod 8 = 7$
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$
$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$
$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$
$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$
$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

To find $11^7 \bmod 13$, we can proceed as follows:

$11^2 = 121 \equiv 4 \pmod{13}$
$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$
$11^7 = 11 \times 4 \times 3 = 132 \equiv 2 \pmod{13}$

Properties of Modular Arithmetic

set of residues, or residue classes (mod n)

Define the set Z_n as the set of nonnegative integers less than n:

$$Z_n = \{0, 1, \ldots, (n-1)\}$$

precise, each integer in Z_n represents a residue class. We can label the residue classes (mod n) as $[0], [1], [2], \ldots, [n-1]$, where

$$[r] = \{a: a \text{ is an integer}, a \equiv r \pmod{n}\}$$

The residue classes (mod 4) are

$[0] = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$
$[1] = \{\ldots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \ldots\}$
$[2] = \{\ldots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \ldots\}$
$[3] = \{\ldots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \ldots\}$

reducing k modulo n.

Finding the smallest nonnegative integer to which k is congruent modulo n

if $(a + b) \equiv (a + c) \pmod{n}$ **then** $b \equiv c \pmod{n}$

$(5 + 23) \equiv (5 + 7) \pmod{8}$; $23 \equiv 7 \pmod{8}$

if $(a \times b) \equiv (a \times c) \pmod{n}$ **then** $b \equiv c \pmod{n}$ **if** a is relatively prime to n

Properties of Modular Arithmetic for Integers in Zn

Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x + w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse ($-w$)	For each $w \in Z_n$, there exists a z such that $w + z = 0 \bmod n$

Table 4.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverses modulo 8

The Euclidean Algorithm

- Simple procedure for determining the greatest common divisor of two positive integers.
- Two integers are relatively prime if their only common positive integer factor is 1

Greatest Common Divisor

- largest integer that divides both a and b
- gcd(0, 0) = 0.

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

$$\gcd(a, b) = \max[k, \text{such that } k|a \text{ and } k|b] \quad \gcd(a, b) = \gcd(|a|, |b|).$$

$$\gcd(60, 24) = \gcd(60, -24) = 12$$

- a and b are relatively prime if gcd(a, b) = 1

Example:

8 and 15 are relatively prime because the positive divisors of 8 are 1, 2, 4, and 8, and the positive divisors of are 1, 3, 5, and 15. So 1 is the only integer on both lists.

Steps

$$\begin{aligned}
a &= q_1 b + r_1 && 0 < r_1 < b \\
b &= q_2 r_1 + r_2 && 0 < r_2 < r_1 \\
r_1 &= q_3 r_2 + r_3 && 0 < r_3 < r_2 \\
&\vdots && \vdots \\
r_{n-2} &= q_n r_{n-1} + r_n && 0 < r_n < r_{n-1} \\
r_{n-1} &= q_{n+1} r_n + 0 \\
d &= \gcd(a, b) = r_n
\end{aligned}$$

Example

To find $d = \gcd(a,b) = \gcd(1160718174, 316258250)$		
$a = q_1 b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$

Dividend	Divisor	Quotient	Remainder
$a = 1160718174$	$b = 316258250$	$q_1 = 3$	$r_1 = 211943424$
$b = 316258250$	$r_1 = 211943434$	$q_2 = 1$	$r_2 = 104314826$
$r_1 = 211943424$	$r_2 = 104314826$	$q_3 = 2$	$r_3 = 3313772$

Euclidean Algorithm Revisited

- For any nonnegative integer a and any positive integer b,
- gcd(a, b) = gcd (b, a mod b)
  - Example: gcd(55, 22) = gcd(22, 55 mod 22) = gcd(22, 11) = 11

Recursive function.

```
Euclid(a,b)
      if (b=0) then return a;
      else return Euclid(b, a mod b);
```

The Extended Euclidean Algorithm

calculate the greatest common divisor but also two additional integers and that satisfy the following equation

$$ax + by = d = \gcd(a, b)$$

x and y will have opposite signs

(4.3), and we assume that at each step i we can find integers x_i and y_i that satisfy $r_i = ax_i + by_i$. We end up with the following sequence.

$$\begin{aligned} a &= q_1 b + r_1 & r_1 &= ax_1 + by_1 \\ b &= q_2 r_1 + r_2 & r_2 &= ax_2 + by_2 \\ r_1 &= q_3 r_2 + r_3 & r_3 &= ax_3 + by_3 \\ &\cdots & &\cdots \end{aligned}$$

$$\begin{aligned} r_{n-2} &= q_n r_{n-1} + r_n & r_n &= ax_n + by_n \\ r_{n-1} &= q_{n+1} r_n + 0 \end{aligned}$$

we can rearrange terms to write

$$r_i = r_{i-2} - r_{i-1} q_i$$

Also, in rows $i - 1$ and $i - 2$, we find the values

$$r_{i-2} = ax_{i-2} + by_{i-2} \quad \text{and} \quad r_{i-1} = ax_{i-1} + by_{i-1}$$

Substituting into Equation (4.8), we have

$$\begin{aligned} r_i &= (ax_{i-2} + by_{i-2}) - (ax_{i-1} + by_{i-1})q_i \\ &= a(x_{i-2} - q_i x_{i-1}) + b(y_{i-2} - q_i y_{i-1}) \end{aligned}$$

But we have already assumed that $r_i = ax_i + by_i$. Therefore,

$$x_i = x_{i-2} - q_i x_{i-1} \quad \text{and} \quad y_i = y_{i-2} - q_i y_{i-1}$$

Let us now look at an example with relatively large numbers to see the power of this algorithm:

To find $d = \gcd(a,b) = \gcd(1160718174, 316258250)$		
$a = q_1 b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$
$r_2 = q_4 r_3 + r_4$	$104314826 = 31 \times 3313772 + 1587894$	$d = \gcd(3313772, 1587894)$
$r_3 = q_5 r_4 + r_5$	$3313772 = 2 \times 1587894 + 137984$	$d = \gcd(1587894, 137984)$
$r_4 = q_6 r_5 + r_6$	$1587894 = 11 \times 137984 + 70070$	$d = \gcd(137984, 70070)$
$r_5 = q_7 r_6 + r_7$	$137984 = 1 \times 70070 + 67914$	$d = \gcd(70070, 67914)$
$r_6 = q_8 r_7 + r_8$	$70070 = 1 \times 67914 + 2156$	$d = \gcd(67914, 2156)$
$r_7 = q_9 r_8 + r_9$	$67914 = 31 \times 2516 + 1078$	$d = \gcd(2156, 1078)$
$r_8 = q_{10} r_9 + r_{10}$	$2156 = 2 \times 1078 + 0$	$d = \gcd(1078, 0) = 1078$
Therefore, $d = \gcd(1160718174, 316258250) = 1078$		

The 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

We use a table to follow the algorithm.

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	

We get gcd (161, 28) = 7, $s = -1$ and $t = 6$. The answers can be tested because we have

$$(-1) \times 161 + 6 \times 28 = 7$$

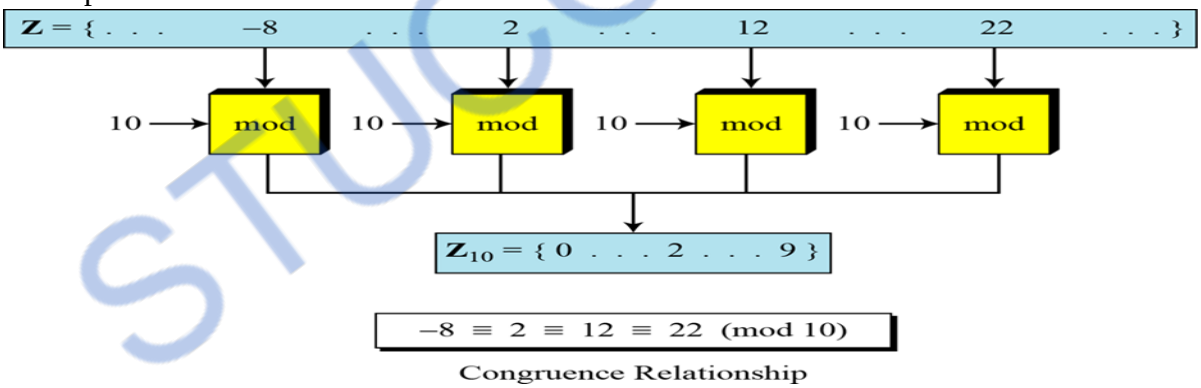
To show that two integers are congruent, we use the congruence operator (≡). For example, we write:

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

$$2 \bmod 10 = 2,$$
$$12 \bmod 10 = 2,$$
$$22 \bmod 10 = 2$$

In modular arithmetic 2, 12 and 22 are called congruent mod10

Concepts of C



Congruence Relationship

INVERSES

To find the inverse of a number relative to an operation

- An additive inverse (relative to an additional operation) and
- A multiplicative inverse (relative to a multiplication inverse)

In Z_n, two numbers a and b are additive inverses of each other if

$$a + b \equiv 0 \pmod{n}$$

Note

In modular arithmetic, each integer has an additive inverse. The sum of an integer and its additive inverse is congruent to 0 modulo n.

Example:

Find all additive inverse pairs in Z_{10}

Solution:

The six pairs of additive inverse are (0,0) (1,9) (2, 8) (3,7) (4,6) and (5,5)

Multiplicative inverse

In Z_n, two numbers a and b are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

Note

In modular arithmetic, an integer may or may not have a multiplicative inverse. When it does, the product of the integer and its multiplicative inverse is congruent to 1 modulo n.

Example

Find the multiplicative inverse of 8 in Z_{10}.

Solution

There is no multiplicative inverse because gcd (10, 8) = 2 ≠ 1. In other words, we cannot find any number between 0 and 9 such that when multiplied by 8, the result is congruent to 1.

Example

Find all multiplicative inverses in Z_{10}.

Solution

There are only three pairs: (1, 1), (3, 7) and (9, 9). The numbers 0, 2, 4, 5, 6, and 8 do not have a multiplicative inverse.

The extended Euclidean algorithm finds the multiplicative inverses of b in Z_n when n and b are given and gcd (n, b) = 1.

The multiplicative inverse of b is the value of t after being mapped to Z_n.

Find the multiplicative inverse of 11 in Z_{26}.

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	–2
2	11	4	3	1	–2	5
1	4	3	1	–2	5	–7
3	3	1	0	5	–7	26
	1	0		–7	26	

The gcd (26, 11) is 1; the inverse of 11 is –7 or 19.

2.4 MATRICES

In cryptography we need to handle matrices

Definition

A matrix of size l x m

m columns

Matrix **A**: l rows

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix}$$

Examples of matrices

$$\begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix}$$
Row matrix

$$\begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}$$
Column matrix

$$\begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix}$$
Square matrix

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$
0

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
I

Operations and Relations

An example of addition and subtraction.

Addition and subtraction of matrices

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

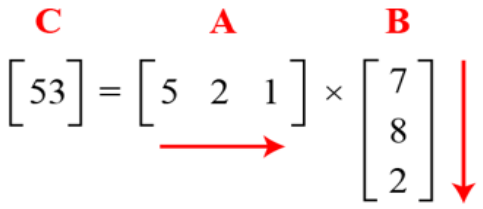
C = A + B

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

D = A − B

Figure shows the product of a row matrix (1 × 3) by a column matrix (3 × 1). The result is a matrix of size 1 × 1.

Figure *Multiplication of a row matrix by a column matrix*



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

DETERMINANT

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

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

Note

The determinant is defined only for a square matrix.

Figure below shows how we can calculate the determinant of a 2 × 2 matrix based on the determinant of a 1 × 1 matrix.

Figure *Calculating the determinant of a 2 ´ 2 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

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

Figure below shows the calculation of the determinant of a 3 × 3 matrix.

Figure *Calculating the determinant of a 3 ´ 3 matrix*

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

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

2.5 Groups, Rings and Fields

Groups, rings, and fields are the fundamental elements of a branch of mathematics known as abstract algebra, or modern algebra. In abstract algebra, we are concerned with sets on whose elements we can operate algebraically; that is, we can combine two elements of the set, perhaps in several ways, to obtain a third element of the set. These operations are subject to specific rules, which define the nature of the set. By convention, the notation for the two principal classes of operations on set elements is usually the same as the notation for addition and multiplication on ordinary numbers. However, it is important to note that, in abstract algebra, we are not limited to ordinary arithmetical operations. All this should become clear as we proceed.

Groups

A **group** G, sometimes denoted by $\{G, \cdot\}$, is a set of elements with a binary operation denoted by $\cdot$ that associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G, such that the following axioms are obeyed:[4]

(A1) Closure:	If a and b belong to G, then $a \cdot b$ is also in G.
(A2) Associative:	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G.
(A3) Identity element:	There is an element e in G such that $a \cdot e = e \cdot a = a$ for all a in G.
(A4) Inverse element:	For each a in G, there is an element a' in G such that $a \cdot a' = a' \cdot a = e$.

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: $a \cdot b = b \cdot a$ for all a, b in G.

The set of integers (positive, negative, and 0) under addition is an abelian group. The set of nonzero real numbers under multiplication is an abelian group. The set S_n from the preceding example is a group but not an abelian group for $n > 2$.

When the group operation is addition, the identity element is 0; the inverse element of a is $-a$; and subtraction is defined with the following rule: $a - b = a + (-b)$.

CYCLIC GROUP We define exponentiation within a group as a repeated application of the group operator, so that $a^3 = a \cdot a \cdot a$. Furthermore, we define $a^0 = e$ as the identity element, and $a^{-n} = (a')^n$, where a' is the inverse element of a within the group. A group G is **cyclic** if every element of G is a power a^k (k is an integer) of a fixed element $a \in G$. 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.

The additive group of integers is an infinite cyclic group generated by the element 1. In this case, powers are interpreted additively, so that n is the nth power of 1.

Rings

A **ring** R, sometimes denoted by $\{R, +, \times\}$, 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. For the case of an additive group, we denote the identity element as 0 and the inverse of a as $-a$.

(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.

In essence, a ring is a set in which we can do addition, subtraction $[a - b = a + (-b)]$, and multiplication without leaving the set.

With respect to addition and multiplication, the set of all n-square matrices over the real numbers is a ring.

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.

Let S be the set of even integers (positive, negative, and 0) under the usual operations of addition and multiplication. S is a commutative ring. The set of all n-square matrices defined in the preceding example is not a commutative ring.

The set Z_n of integers $\{0, 1, \ldots, n-1\}$, together with the arithmetic operations modulo n, is a commutative ring (Table 4.3).

Next, we define 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 $b = 0$.

Let S be the set of integers, positive, negative, and 0, under the usual operations of addition and multiplication. S is an integral domain.

Fields

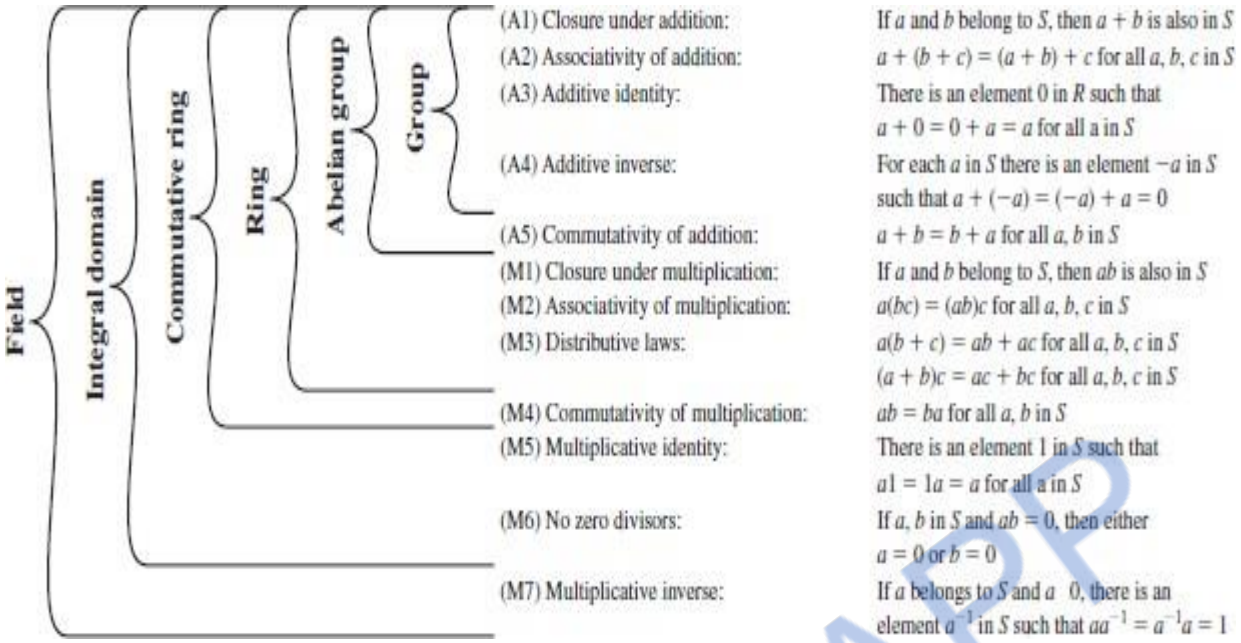
A **field** F, sometimes denoted by $\{F, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in F the following axioms are obeyed.

(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 $aa^{-1} = (a^{-1})a = 1$.

In essence, a field is a set in which we can do addition, subtraction, multiplication, and division without leaving the set. Division is defined with the following rule: $a/b = a(b^{-1})$.

Familiar examples of fields are the rational numbers, the real numbers, and the complex numbers. Note that the set of all integers is not a field, because not every element of the set has a multiplicative inverse; in fact, only the elements 1 and –1 have multiplicative inverses in the integers.



Summarize the Axioms that defines Groups, Rings and Fields

2.6 FINITE FIELDS OF THE FORM GF(p)

The finite field of order p^n is generally written $GF(p^n)$; GF stands for Galois field, in honor of the mathematician who first studied finite fields. Two special cases are of interest for our purposes. For $n = 1$, we have the finite field $GF(p)$; this finite field has a different structure than that for finite fields with $n > 1$ and is studied in this section. In Section 4.7, we look at finite fields of the form $GF(2^n)$.

Finite Fields of Order p

For a given prime, p, we define the finite field of order p, $GF(p)$, as the set Z_p of integers $\{0, 1, \ldots, p - 1\}$ together with the arithmetic operations modulo p.

Recall that we showed in Section 4.3 that the set Z_n of integers $\{0, 1, \ldots, n - 1\}$, together with the arithmetic operations modulo n, is a commutative ring (Table 4.3). We further observed that any integer in Z_n has a multiplicative inverse if and only if that integer is relatively prime to n [see discussion of Equation (4.5)].[7] If n is prime, then all of the nonzero integers in Z_n are relatively prime to n, and therefore there exists a multiplicative inverse for all of the nonzero integers in Z_n. Thus, for Z_p we can add the following properties to those listed in Table 4.3:

Multiplicative inverse (w^{-1})	For each $w \in Z_p$, $w \neq 0$, there exists a $z \in Z_p$ such that $w \times z \equiv 1 \pmod p$

The simplest finite field is GF(2). Its arithmetic operations are easily summarized:

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

w	$-w$	w^{-1}
0	0	—
1	1	1

Inverses

In this case, addition is equivalent to the exclusive-OR (XOR) operation, and multiplication is equivalent to the logical AND operation.

2.7 Symmetric Key Ciphers

The Feistel Cipher Structure

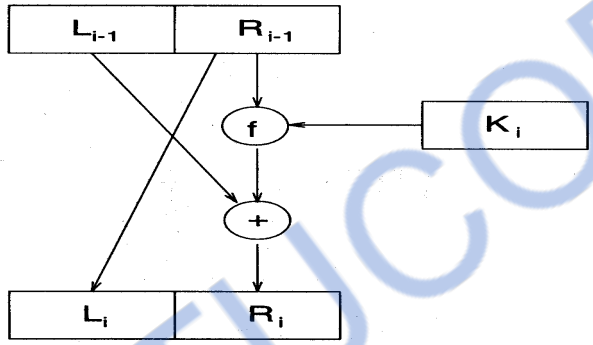
Feistel proposed a structure which alternates substitution and permutation. This follows the concept given by claude Shannon to produce cipher that alternates confusion and diffusion.

Diffusion: making each plaintext bit affect as many ciphertext bits as possible.

Confusion: making the relationship between the encryption key and the ciphertext as complex as possible.

- Input: a data block and a key
- Partition the data block into two halves L and R.
- **Go through a number of rounds**.
- In each round,
  - R does not change.
  - L goes through an operation that depends on R and a round key derived from the key.

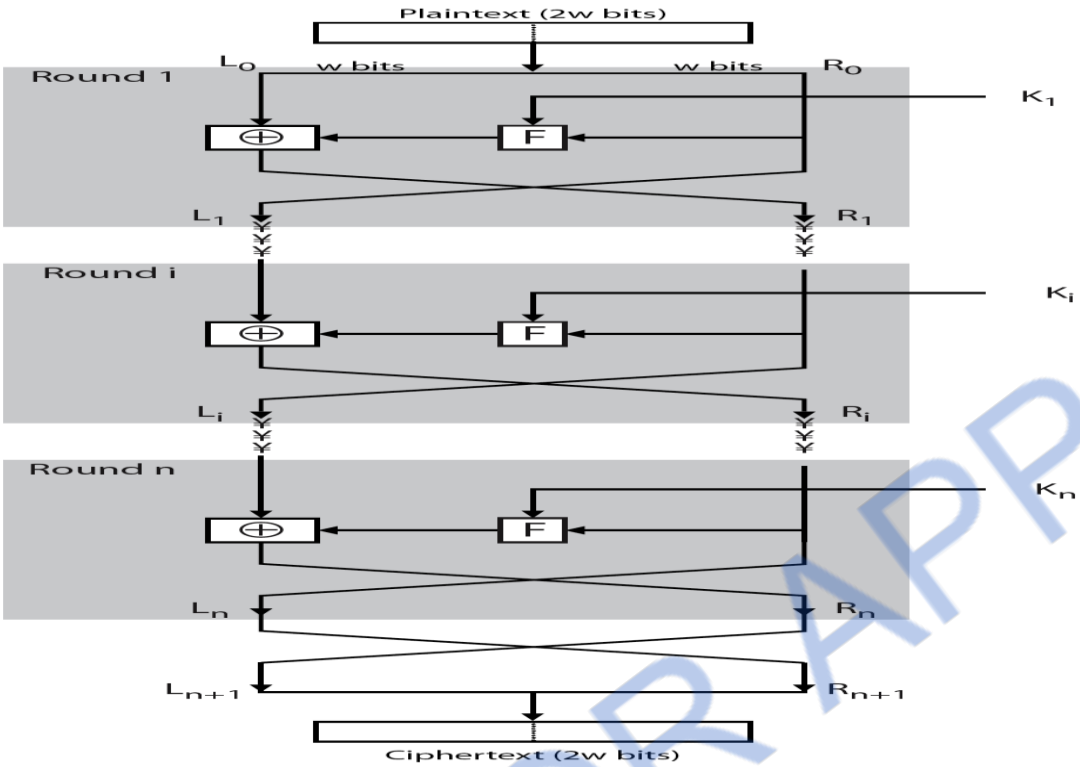
Feistel Cipher (Single round)



$L_i = R_{i-1}$

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

The Feistel Cipher Structure



2.8 SIMPLE DES

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977.

Encryption

Takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8-bit of cipher.

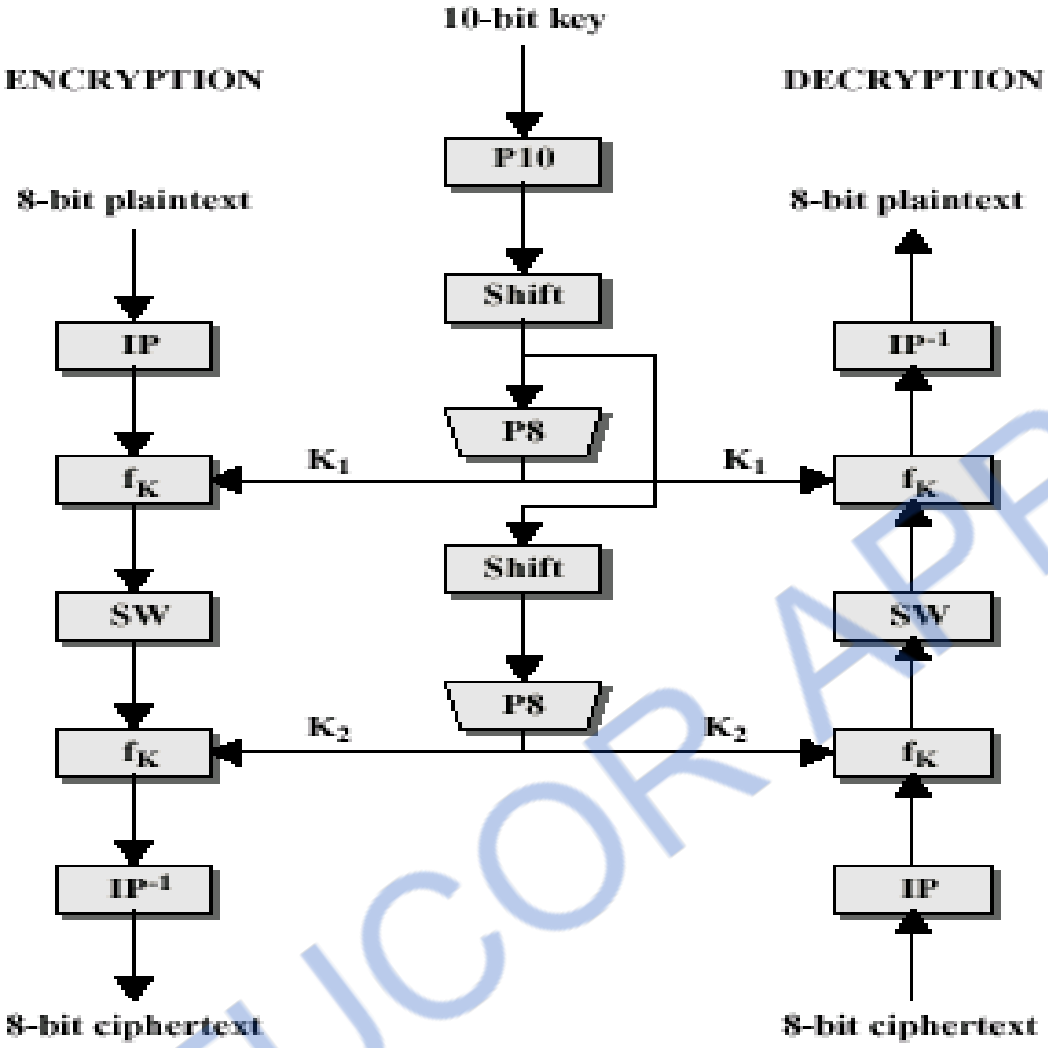
Decryption

Takes an 8-bit block of cipher and the same 10-bit key as input and produces an 8-bit of original plaintext.

- Both substitution and transposition operations are used
- It is a complex, multi-phase algorithm

Five Functions to Encrypt

- IP: Initial permutation
- f_k: Key-dependent scrambler ((complex) function))
  - Use a 8-bit key
  - Perform both permutation and substitution
- SW (simple permutation function)
  - Swap the two halves of data
- f_k again (different key)
- IP^{-1}: Inverse permutation



S-DES Algorithm

We can concisely express the encryption algorithm as a composition of function: $IP^{-1} \circ f_{k2} \circ SW \circ f_{k1} \circ IP$

OR AS:

- Cipher = $IP^{-1}(f_{k2}(SW(f_{k1}(IP(plaintext)))))$
- $K_1 = P8(Shift(P10(key)))$
- $K_2 = P8(Shift(Shift(P10(key))))$
- Plaintext = $IP^{-1}(f_{k1}(SW(f_{k2}(IP(ciphertext)))))$

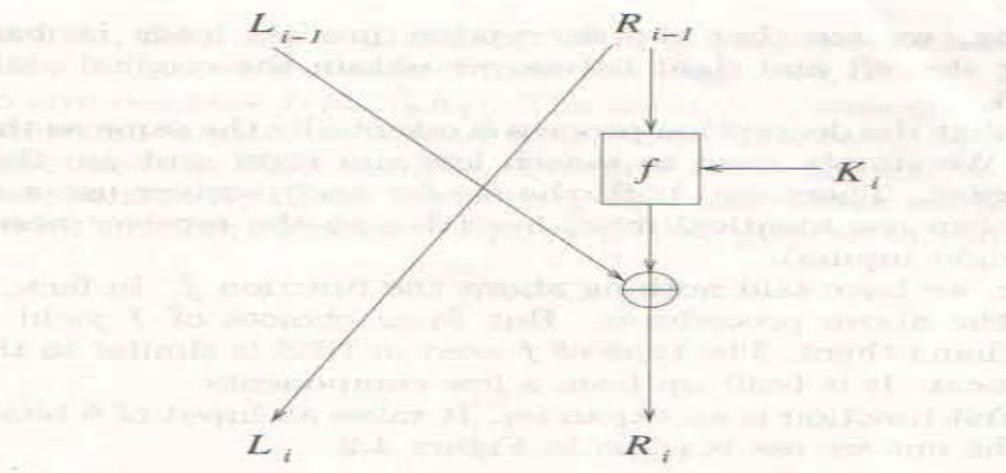


Fig: One round of Feistel system

- Assume the full message consists of only one block.
- The message has 12 bits and is written in the form $L_0 R_0$, where L_0 consists of the first 6 bits and R_0 consists of the last 6 bits.
- The key K has 9 bits. The i^{th} round of the algorithm transforms an input $L_{i-1} R_{i-1}$ to the output $L_I R_I$ using an 8-bit key K_I, derived from K.
- The main part of the encryption process is a function $F(R_{I-1}, K_Z)$ that takes a 6-bit input and an 8-bit input K_I and produces a 6-bit output.
- The output for the i^{th} round is defined as follows:
- $Li = R_{I-1}$ and $R_I = L_{I-1} \Phi F(R_{I-1}, K_Z)$
- This operation is performed for a certain number of rounds, say n, and produces the ciphertext $L_n R_n$.
- The first step takes $R_n L_u$ and gives the output $[L_n]\ [R_n \circledR f(L_n . K_n)]$.

Encryption

8-Bit Plaintext: Make up by sender

1	1	1	1	0	0	1	1

IP: Initial Permutation (constant)

2	6	3	1	4	8	5	7

IP^{-1}: Inversed Permutation (constant)

4	1	3	5	7	2	8	6

Complex function f_k:

E/P: Expansion/Permutation Rule (constant)

4	1	2	3	2	3	4	1

The first function is an expander.

It takes an input of 6 bits and outputs 8 bits.

This means that the first input bit yields the first output bit, the third input bit yields both the fourth and the sixth output bits, etc.

For example, 011001 is expanded to 01010101.

The rep is

n_4 n_1 n_2 n_3

n_2 n_3 n_4 n_1

P4: Permutation 4 (constant)

2	4	3	1

The main components are called S-boxes. We use two:

S1 = 101 010 001 110 011 100 111 000

001 100 110 010 000 111 101 011

S2 = 100 000 110 101 111 001 011 010

101 011 000 111 110 010 001 100 •

The input for an S-box has 4 bits.

The first bit specifies which row will be used: 0 for the first row, 1 for the second. The other 3 bits represent a binary number that specifies the column: 000 for the first column, 001 for the second, ..., Ill for the last column.

The output for the S-box consists of the three bits in the specified location.

The key *K* consists of 9 bits. The key *Ki* for the ith round of encryption is obtained by using 8 bits of K, starting with the z'th bit.

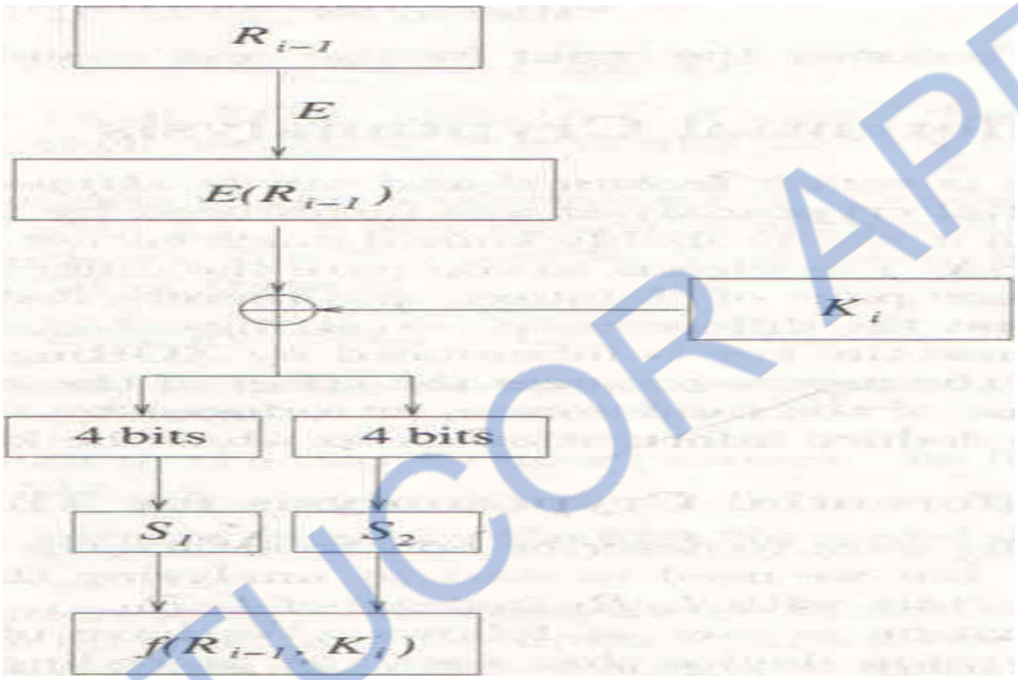


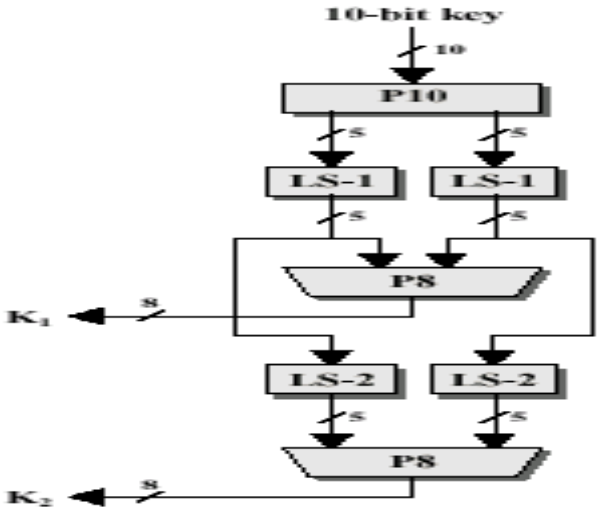
Fig: The function f($R_{i-1,Ki}$)

We can now describe *F(RI*-1, /K i). The input R_{i-1} consists of 6 bits. The expander function is used to expand it to 8 bits.

The result is XORed with K, to produce another 8-bit number. The first 4 bits are sent to S1. and the last 4 bits are sent to S2.

Each S-box outputs 3 bits, which are concatenated to form a 6-bit. number. This is *F{FB-I, KI).*

Key Generation



Example of Encryption

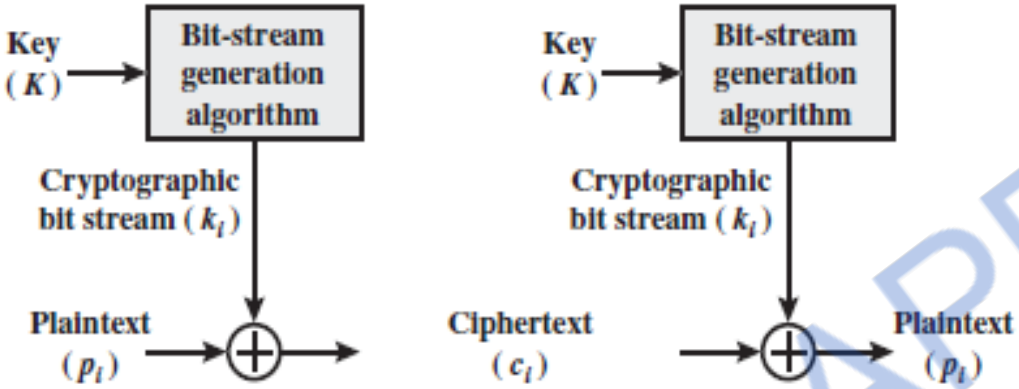
X:8-bit Plaintext	1	1	1	1	0	0	1	1
IP8: Initial permutation vector	2	6	3	1	4	8	5	7
Permutation of X	1	0	1	1	1	1	0	1
Splitting into L0,R0	1	0	1	1	1	1	0	1
E/P 8: Expansion permutation of R0	4	1	2	3	2	3	4	1
EP(0): Expanded R0	1	1	1	0	1	0	1	1
K1: Key 1	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**
EP(R0) xor K1	0	1	0	0	1	1	1	1

Analysis:

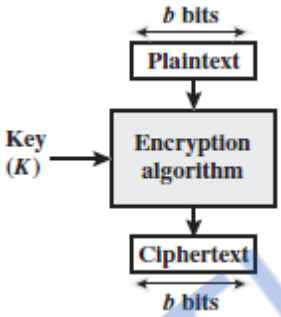
Brute force is feasible.

2.9 Block Cipher Principles

A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenere cipher and the Vernam cipher.

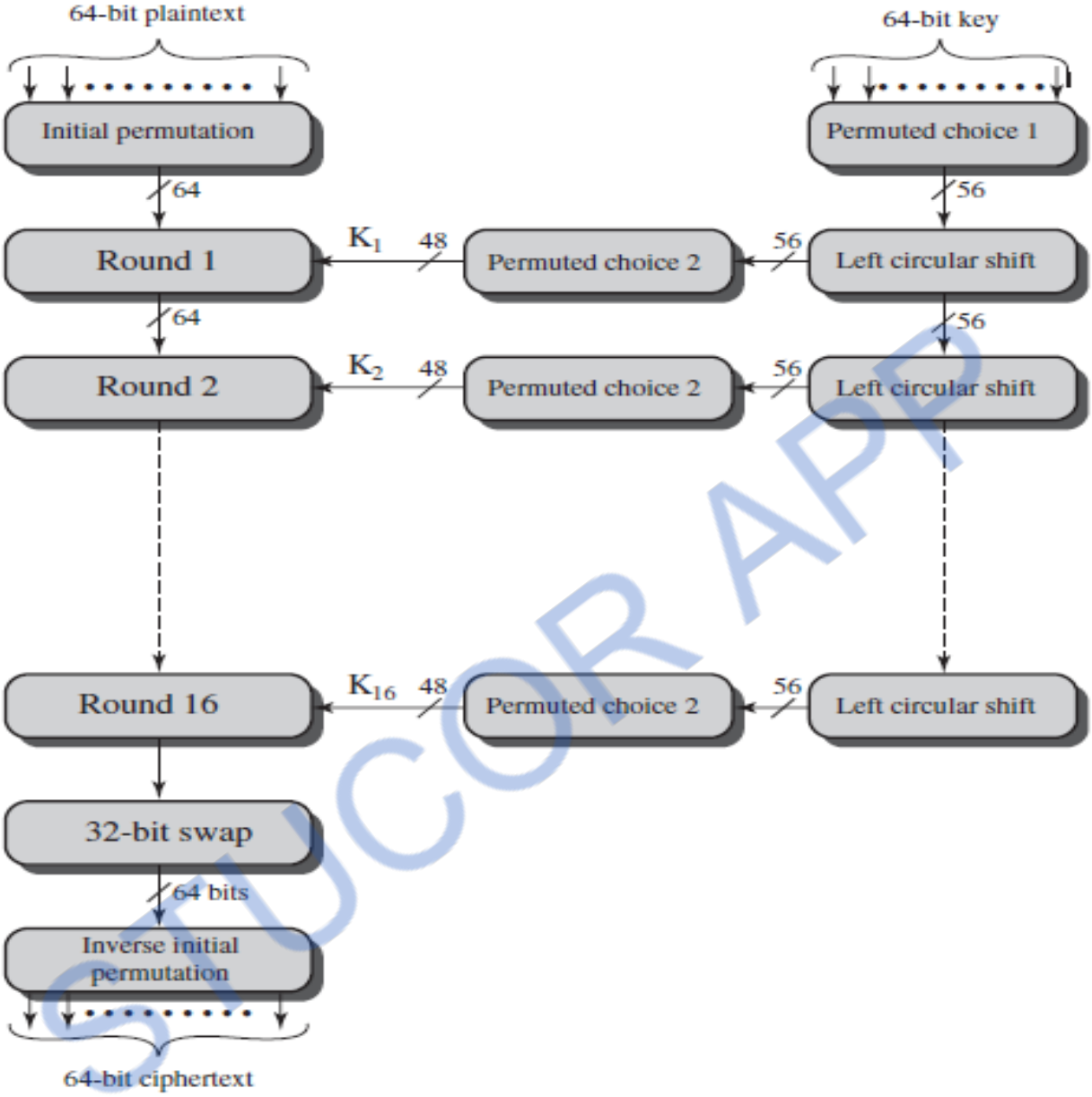


ENCYPTION



DECRYPTION

2.10 DATA ENCYPTION STANDARD



Block Diagram of DES

As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.

Input of 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the **permuted input.** This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**.

Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a **subkey (K_i)** is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

(a) Initial Permutation (IP)

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

(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

Input PlainText M

$$\begin{matrix} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} \\ M_{17} & M_{18} & M_{19} & M_{20} & M_{21} & M_{22} & M_{23} & M_{24} \\ M_{25} & M_{26} & M_{27} & M_{28} & M_{29} & M_{30} & M_{31} & M_{32} \\ M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} & M_{39} & M_{40} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} & M_{48} \\ M_{49} & M_{50} & M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} \\ M_{57} & M_{58} & M_{59} & M_{60} & M_{61} & M_{62} & M_{63} & M_{64} \end{matrix}$$

Where M_i is a binary digit. Then the permutation X=IP(M)

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

The inverse permutation Y = IP-1(X) = IP-1(IP(M))

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).

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

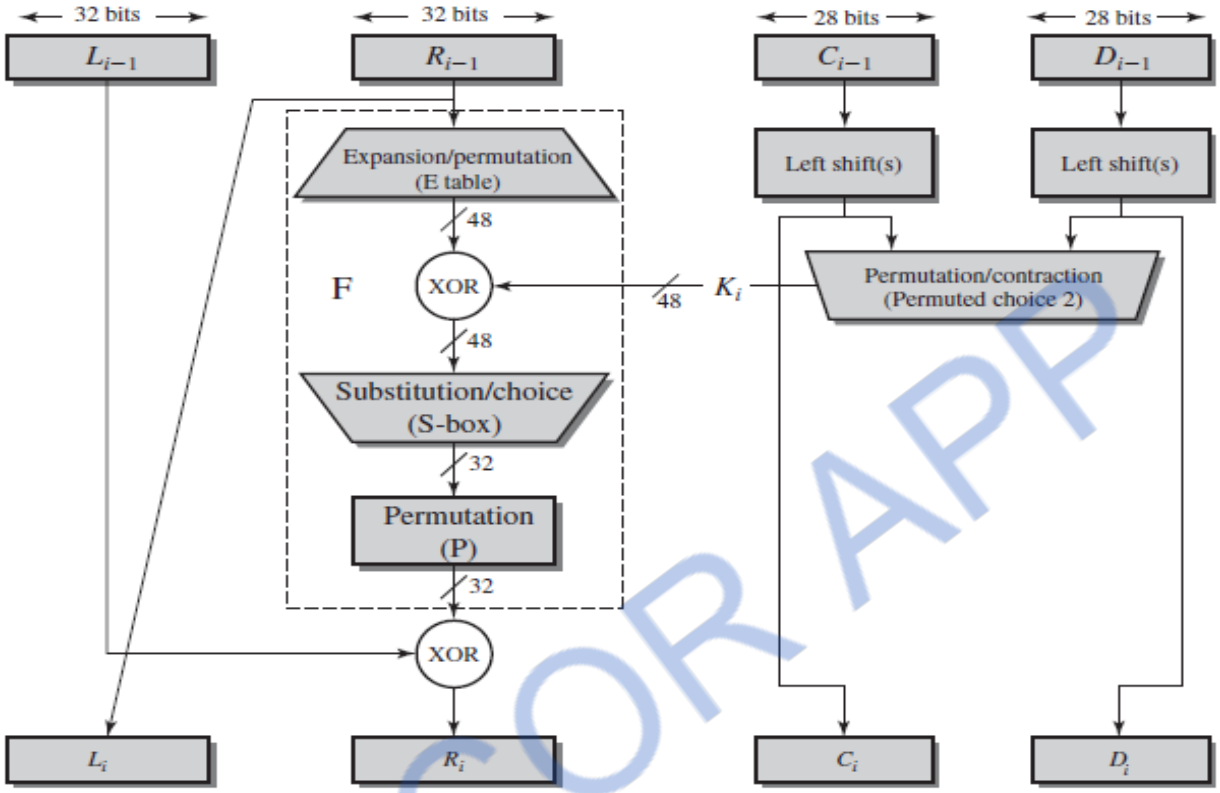
The round key Ki is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits. The resulting 48 bits are XORed with Ki. This 48-bit result passes through a substitution function that produces a 32-bit output.

The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.

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. For example, in S1, for input 011001, 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.

the input word is
... defghi hijklm lmnopq ...
this becomes
... efgh ijkl mnop ...



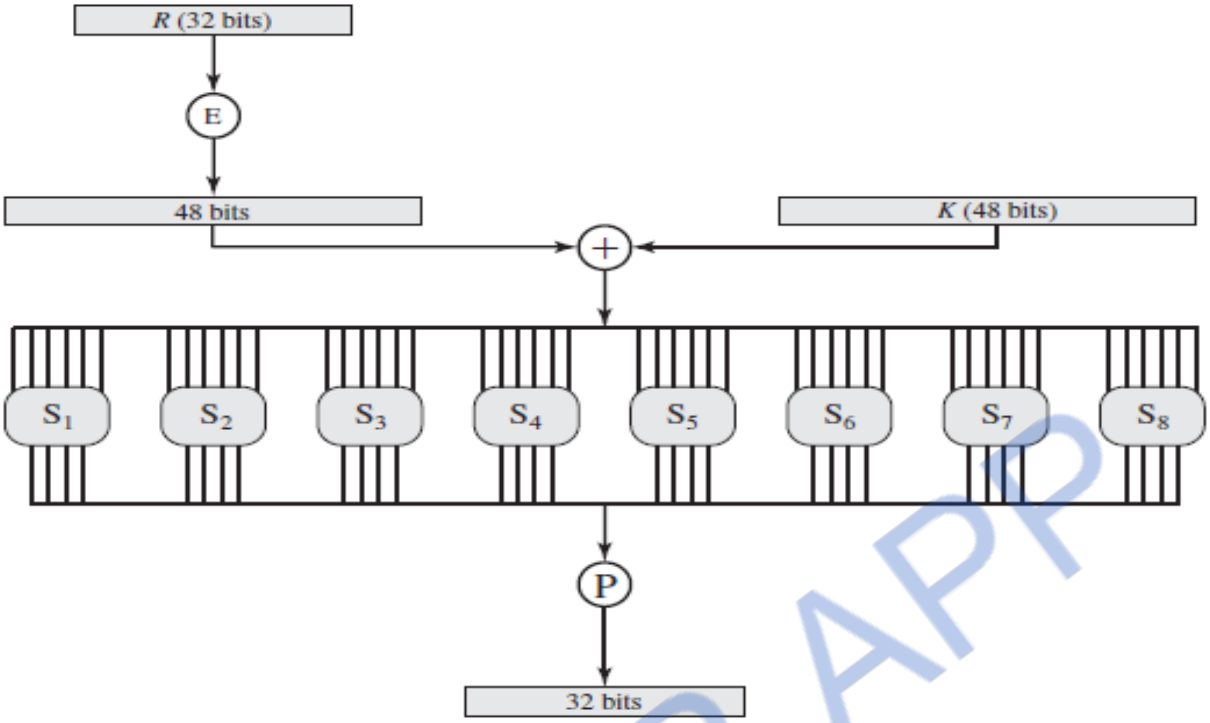
Round Function

Key Generation

64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64. The key is first subjected to a permutation governed by a
table labeled Permuted Choice One. The resulting 56-bit key is then treated as

two 28-bit quantities, labeled C0 and D0. At each round, Ci-1 and Di-1 are separately subjected to a circular left shift or (rotation) of 1 or 2 bits.

These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two, which produces a 48-bit output that serves as input to the function F(Ri-1, Ki).



SUBSTITUTION BOX

DES Decryption

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

The Avalanche Effect

That a small change in either the plaintext or the key should produce a significant change in the cipher text. a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the cipher text. This is referred to as the avalanche effect.

(a) 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

(b) Permuted Choice One (PC-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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

Round Number	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

Substitution

Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.

Permutation

A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

The Strength of DES

The Use of 56-Bit Keys

With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 * 1016 keys.

The Nature of the DES Algorithm

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public,

There is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes. This assertion is tantalizing, and over the years a number of regularities and unexpected behaviors of the S-boxes have been discovered.

Timing Attacks

A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts. A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.

DES appears to be fairly resistant to a successful timing attack but suggest some avenues to explore. Although this is an interesting line of attack, it so far appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES.

2.11 DIFFERENTIAL AND LINEAR CRYPTANALYSIS

Differential Cryptanalysis

The differential cryptanalysis attack is complex; provides a complete description. 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. The original plaintext
block m to consist of two halves m0 and m1 . 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 subkey for this round. So, at each round, only one new 32-bit block is created. Intermediate message halves are related as

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

In differential cryptanalysis, we start with two messages, m and m', with a known XOR difference $\Delta m = m \oplus m'$, and consider the difference between the intermediate message halves: $\Delta m_i = m_i \oplus m'_i$. Then we have

$$\begin{aligned} \Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)] \end{aligned}$$

suppose that many pairs of inputs to f with the same difference yield the same output difference if the same subkey is used. if a number of such differences are determined, it is feasible to determine the subkey used in the function F.

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 known plaintexts, as compared to 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.
For a cipher with -bit plaintext and cipher text blocks and an -bit key, let the plaintext block be labelled P[1],....p[n], the cipher text block C[1],.....,C[n], and the key K[1],.....,K[m] .Then define

$$A[i, j, \ldots, k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

The objective of linear cryptanalysis is to find an effective linear equation of the form

$$P[\alpha_1, \alpha_2, \ldots, \alpha_a] \oplus C[\beta_1, \beta_2, \ldots, \beta_b] = K[\gamma_1, \gamma_2, \ldots, \gamma_c]$$

2.12 BLOCK CIPHER DESIGN PRINCIPLES

DES Design Criteria

The criteria used in the design of DES, focused on the design of the S-boxes and on the P function that takes the output of the S-boxes .The criteria for the S-boxes are as follows.
1. No output bit of any S-box should be too close a linear function of the input bits. Specifically, if we select any output bit and any subset of the six input bits, the fraction of inputs for which this output bit equals the XOR of these input bits should not be close to 0 or 1, but rather should be near 1/2.

2. Each row of an S-box (determined by a fixed value of the leftmost and rightmost input bits) should include all 16 possible output bit combinations.

3. If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits.

4. If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.

5. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.

6. For any nonzero 6-bit difference between inputs, no more than eight of the
32 pairs of inputs exhibiting that difference may result in the same output difference.

7. This is a criterion similar to the previous one, but for the case of three S-boxes.

The criteria for the permutation P are as follows.

1. The four output bits from each S-box at round are distributed so that two of them affect (provide input for) “middle bits” of round (i+1) and the other two affect end bits. The two middle bits of input to an S-box are not shared with adjacent S-boxes. The end bits are the two left-hand bits and the two right-hand bits, which are shared with adjacent S-boxes.

2. The four output bits from each S-box affect six different S-boxes on the next round, and no two affect the same S-box.

3. For two S-boxes j,k if an output bit from Sj affects a middle bit of Sk on the next round, then an output bit from Sk cannot affect a middle bit of Sj . This implies that, for j=k , an output bit from Sj must not affect a middle bit of Sj.

Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F. In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack. The differential cryptanalysis attack requires 255.1 operations, whereas brute force 255 requires. If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search.

Design of Function F

It must be difficult to "unscramble" the substitution performed by F. One obvious criterion is that F be nonlinear, as we discussed previously. The more nonlinear F, the more difficult any type of cryptanalysis will be. There are several measures of nonlinearity, which are beyond the scope of this book. In rough terms, the more difficult it is to approximate F by a set of linear equations, the more nonlinear F is.

S-Box Design

- Random: Use some pseudorandom number generation or some table of random digits to generate the entries in the S-boxes. This may lead to boxes with undesirable characteristics for small sizes but should be acceptable for large S-boxes.
- Random with testing: Choose S-box entries randomly, then test the results against various criteria, and throw away those that do not pass.
- Human-made: This is a more or less manual approach with only simple mathematics to support it. It is apparently the technique used in the DES design. This approach is difficult to carry through for large S-boxes.
- Math-made: Generate S-boxes according to mathematical principles. By using mathematical construction, S-boxes can be constructed that offer proven security against linear and differential cryptanalysis, together with good diffusion.

Key Schedule Algorithm

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 working back to the main key. The key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

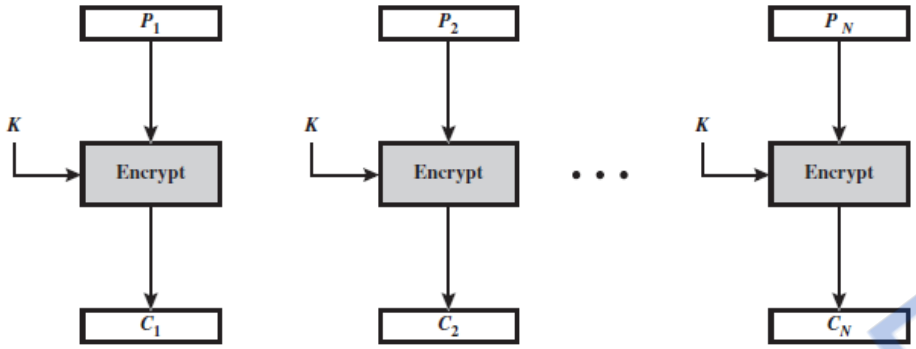
2.13 BLOCK CIPHER MODES OF OPERATION

A block cipher takes a fixed-length block of text of length bits and a key as input and produces a -bit block of ciphertext. If the amount of plaintext to be encrypted is greater than b bits, then the block cipher can still be used by breaking the plaintext up into b-bit blocks.

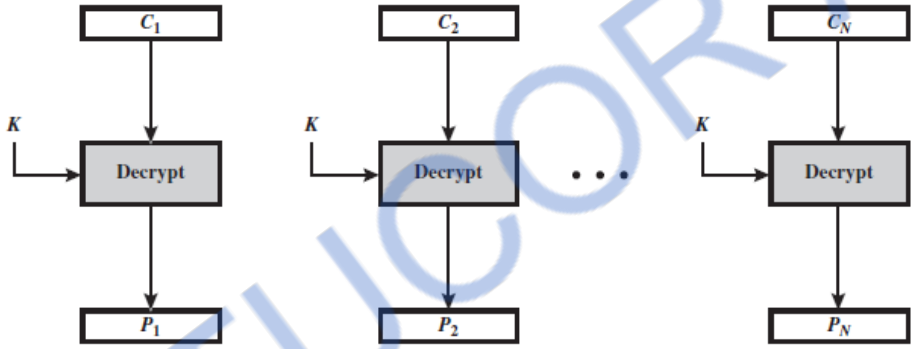
Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	Secure transmission of single values.
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	General-purpose block oriented
		Authentication
Cipher Feedback (CFB)	Input is processed 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
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

ELECTRONIC CODEBOOK (ECB)

In which plaintext is handled one block 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 -bit block of plaintext.



(a) Encryption



(b) Decryption

$C_j = E(K, P_j)$ j = 1,.....,N
$P_j = D(K, C_j)$ j = 1,.....,N

CIPHER BLOCK CHAINING MODE

The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block.

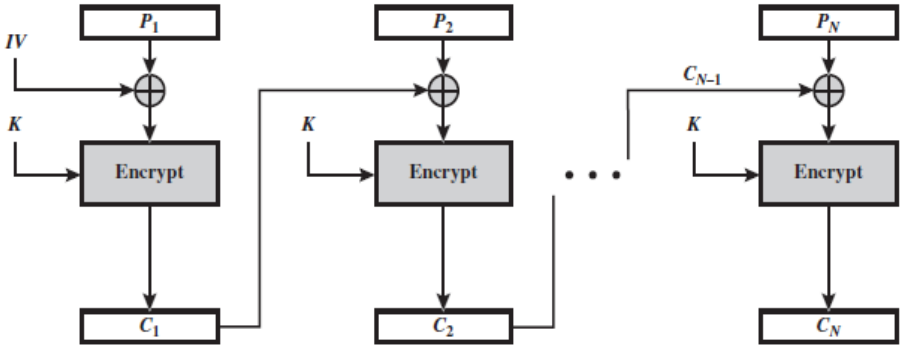
$$C_j = E(K, [C_{j-1} \oplus P_j])$$

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

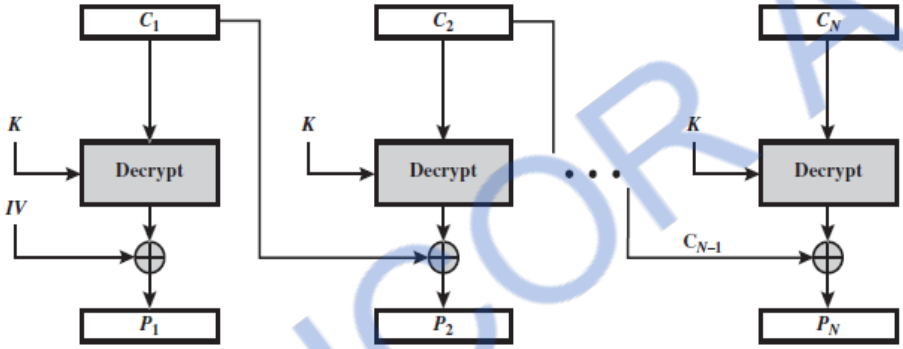
$$D(K, C_j) = C_{j-1} \oplus P_j$$

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

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



(a) Encryption



(b) Decryption

The IV must be known to both the sender and receiver but be unpredictable by a third party. In particular, for any given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV. For maximum security, the IV should be protected against unauthorized changes. This could be done by sending the IV using ECB encryption.

CIPHER FEEDBACK MODE

For AES, DES, or any block cipher, encryption is performed on a block of *b* bits. In the case of DES, *b* = 64 and in the case of AES, *b* = 128.
The CFB scheme, the unit of transmission is *s* bits; a common value is *s* = 8. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than blocks of b bits, the plaintext is divided into segments of s bits

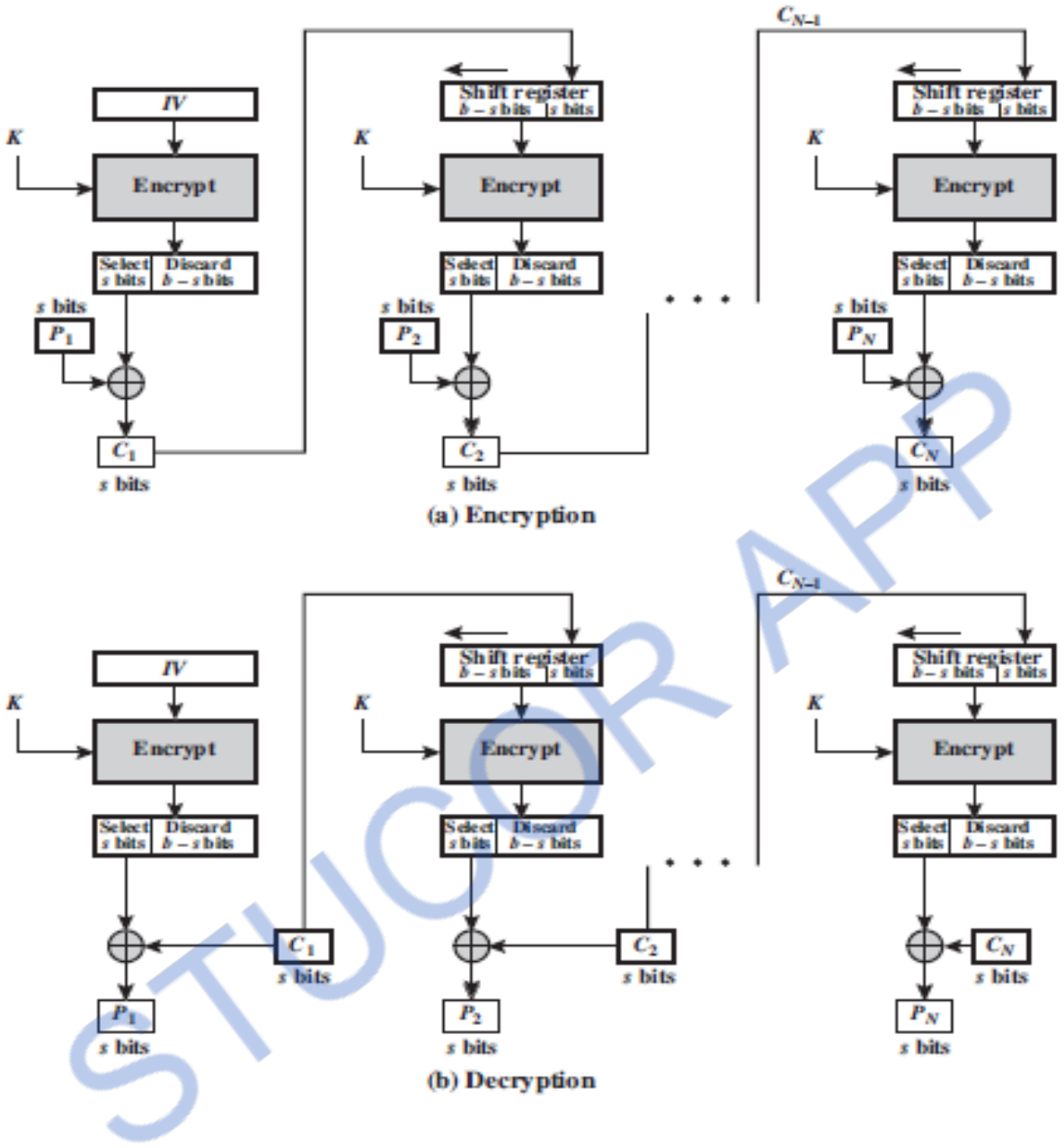
The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV).The leftmost (most significant) bits of the output of the encryption function are XORed with the first segment of plaintext P_1 to produce the first unit of ciphertext C_2, which is then transmitted. In addition, the contents of the shift register are shifted left by bits, and C_1 is placed in the rightmost (least significant) bits of the shift register. This process continues until all plaintext units have been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the *encryption* function that is used, not the decryption function.

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

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

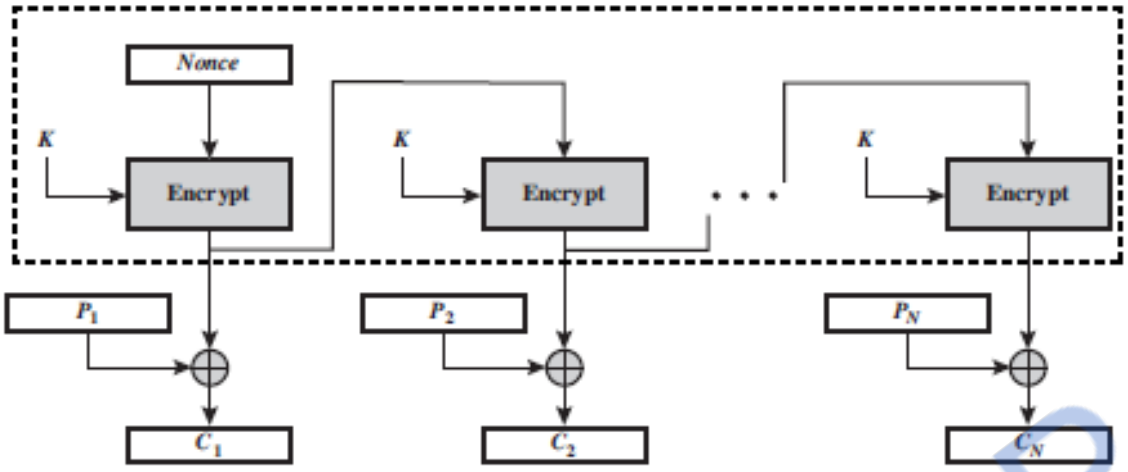
$$\begin{aligned} I_1 &= IV \\ I_j &= \text{LSB}_{b-s}(I_{j-1}) \,\|\, C_{j-1} \quad j = 2, \ldots, N \\ O_j &= \text{E}(K, I_j) \quad j = 1, \ldots, N \\ P_j &= C_j \oplus \text{MSB}_s(O_j) \quad j = 1, \ldots, N \end{aligned}$$



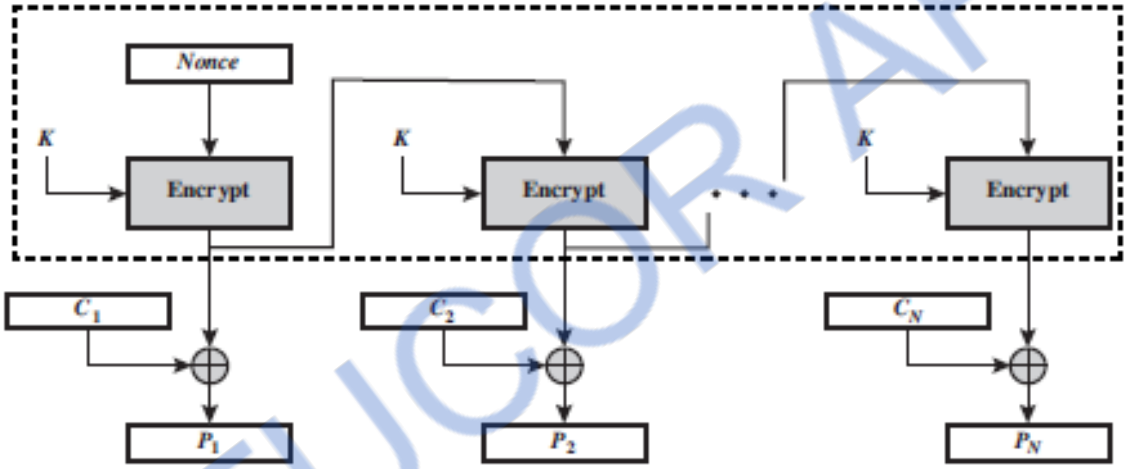
OUTPUT FEEDBACK MODE

The **output feedback** (OFB) mode is similar in structure to that of CFB. 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 shift register. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on an b-bit subset.



(a) Encryption



(b) Decryption

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

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

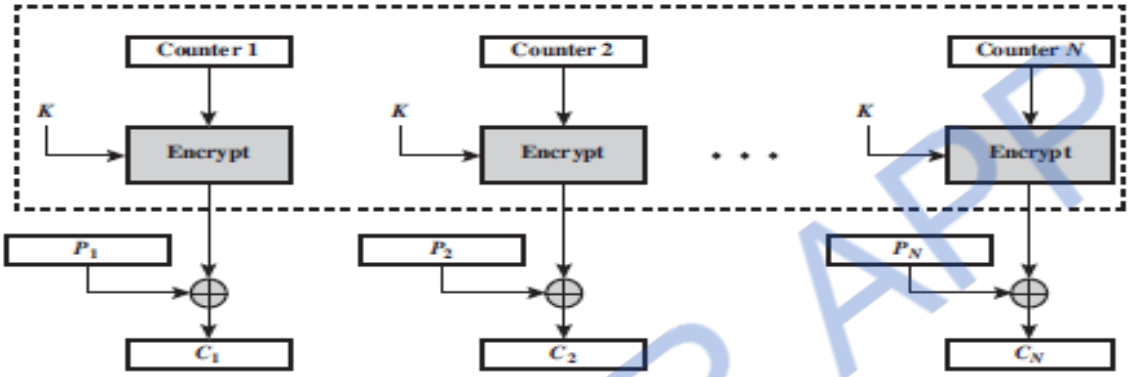
COUNTER MODE

The **counter** (CTR) mode has increased recently with applications to ATM (asynchronous transfer mode) network security and IP sec (IP security). The counter is initialized to some value and then incremented by

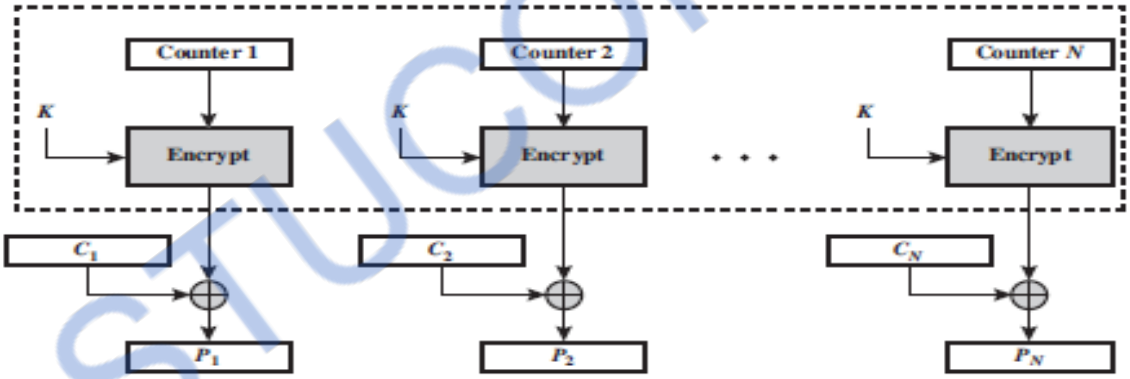
1 for each subsequent block (modulo 2^b where is the block size). For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block. Thus, the initial counter value must be made available for decryption.

$$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1 \qquad P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$$

$$C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)] \qquad P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$$



(a) Encryption



(b) Decryption

Advantages of CTR Mode:

Hardware efficiency: Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext. For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block. This limits the maximum throughput of the algorithm to the reciprocal of the time for one
execution of block encryption or decryption. In CTR mode, the throughput is
only limited by the amount of parallelism that is achieved.

Software efficiency: Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions, can be effectively utilized.

Preprocessing: The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions.

Random access: The th block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block C_i cannot be computed until the i-1 prior block are computed. There may be applications in which a ciphertext is stored and it is desired to decrypt just one block; for such applications, the random access feature is attractive.

Provable security: It can be shown that CTR is at least as secure as the other modes discussed in this section.

Simplicity: Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES. In addition, the decryption key scheduling need not be implemented.

2.15 ADVANCED ENCRYPTION STANDARD

AES is a block cipher intended to replace DES for commercial applications. It uses a 128-bit block size and a key size of 128, 192, or 256 bits.
Each full round consists of four separate functions: byte substitution, permutation, arithmetic operations, over a finite field, and XOR with a key.

AES Parameters

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	10	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

Detailed Structure

1. AES instead processes the entire data block as a single matrix during each round using substitutions and permutation.

2. 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.

3. 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 over
• AddRoundKey: A simple bitwise XOR of the current block with a portion
of the expanded key.

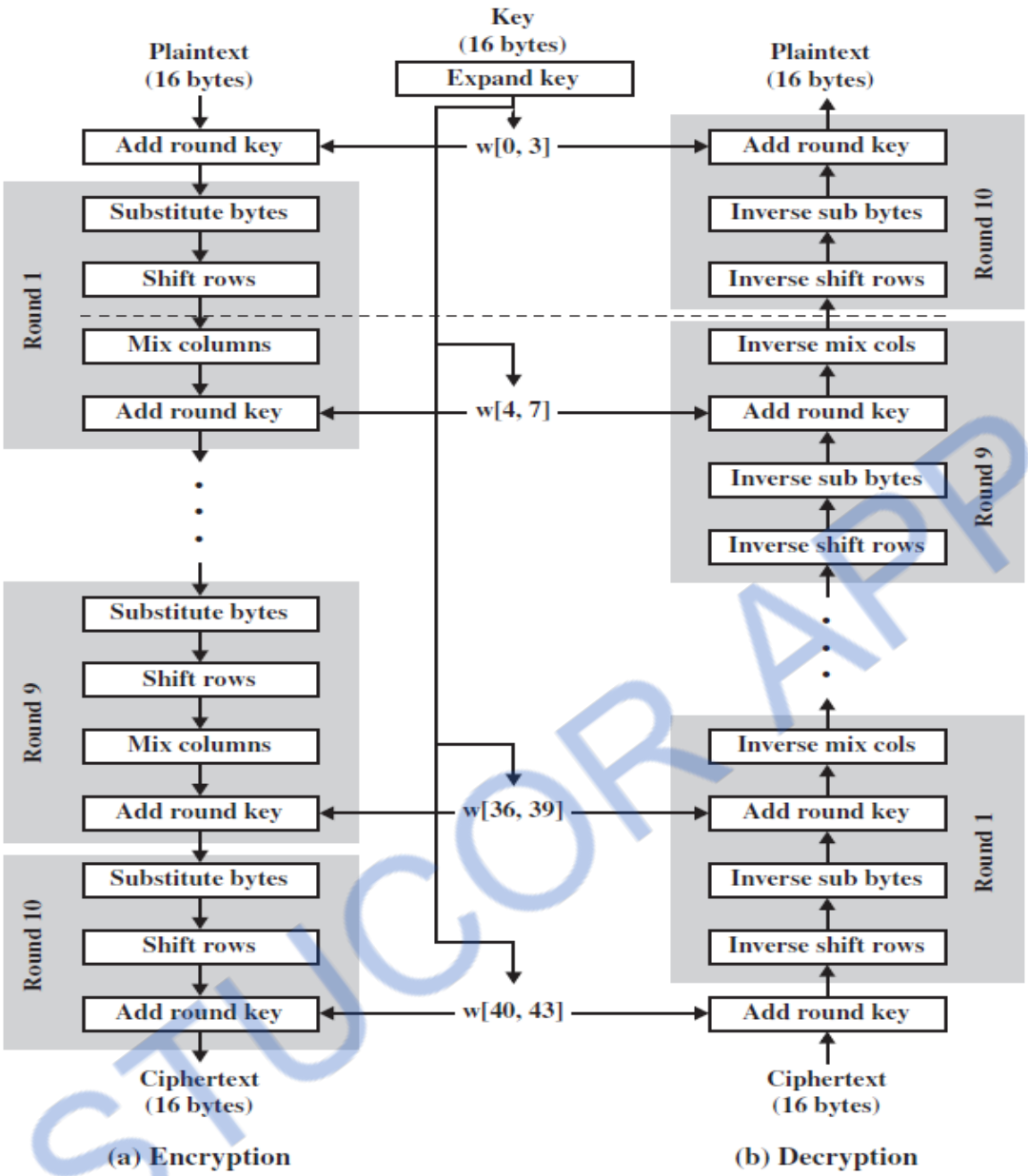
4. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.

5. Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.

6. The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on.This scheme is both efficient and highly secure.

7. Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm.
For the AddRoundKey stage, the inverse is achieved by XORing the same round key to the block.

$A \oplus B \oplus B = A.$



BLOCK DIAGRAM OF AES

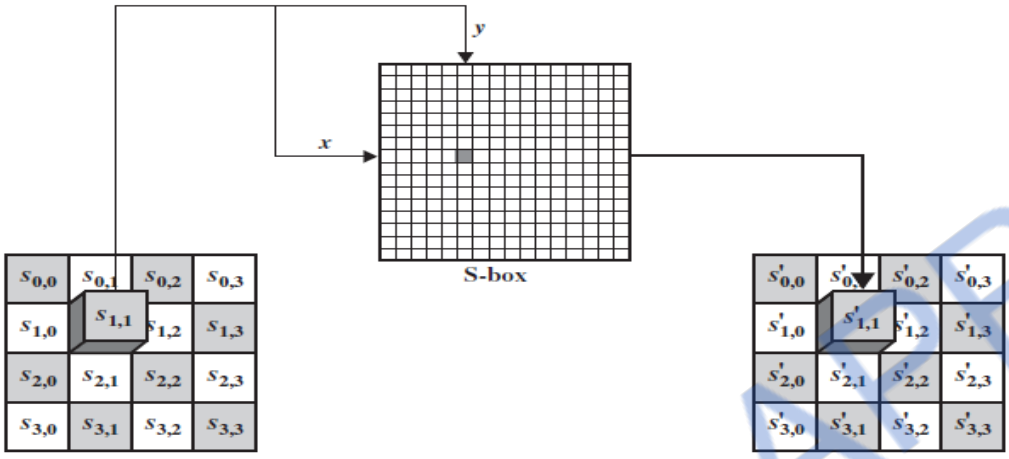
8. The decryption algorithm makes use of the expanded key in reverse order. The decryption algorithm is not identical to the encryption algorithm.

9. The final round of both encryption and decryption consists of only three Stages. This is a consequence of the particular structure of aes and is required to make the cipher reversible.

Substitute Bytes Transformation

The forward substitute byte transformation, called SubBytes, is a simple table lookup. AES defines a 16X16 matrix of byte values, called an S-box. That contains a permutation of all

possible 256 8-bit values. Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9, column 5



EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

SUBSTITUTION BYTE TRANSFORMAION

S BOX

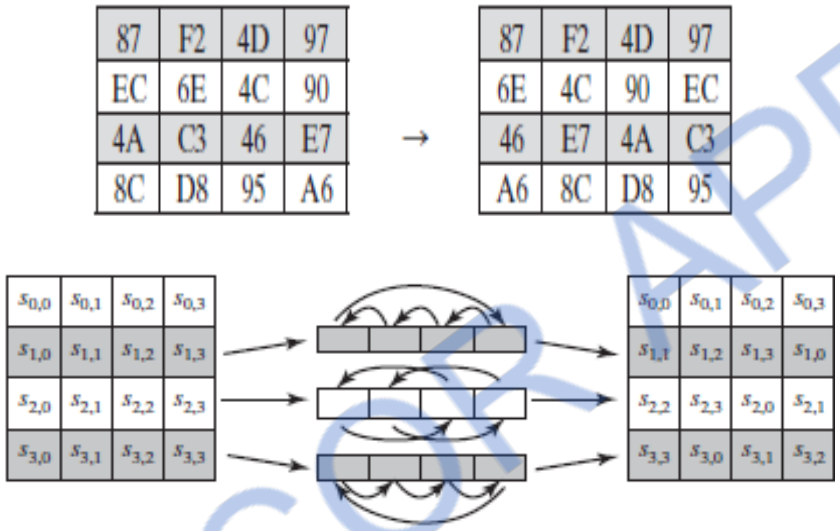
x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

INVERSE BOX

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

ShiftRows Transformation

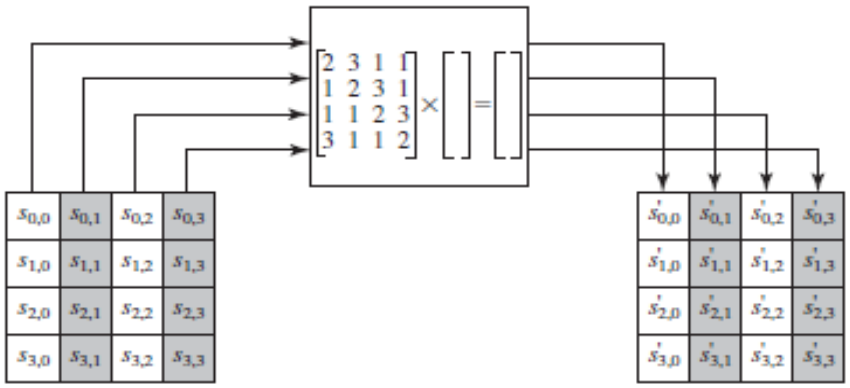
The first row of State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed. The inverse shift row transformation, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.



SHIFT ROW TRANSFORMATION

MixColumn Transformation

Each byte of a column is mapped into a new value that is a function of all four bytes in that column. Multiplication of a value by (i.e., by {02}) can be implemented as a 1-bit left shift followed by a conditional bitwise XOR with (0001 1011) if the leftmost bit of the original value (prior to the shift) is 1. Thus, to verify the MixColumns transformation on the first column



$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned}$$

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

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

$$\begin{aligned} (\{02\} \cdot \{87\}) \oplus (\{03\} \cdot \{6E\}) \oplus \{46\} \oplus \{A6\} &= \{47\} \\ \{87\} \oplus (\{02\} \cdot \{6E\}) \oplus (\{03\} \cdot \{46\}) \oplus \{A6\} &= \{37\} \\ \{87\} \oplus \{6E\} \oplus (\{02\} \cdot \{46\}) \oplus (\{03\} \cdot \{A6\}) &= \{94\} \\ (\{03\} \cdot \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \cdot \{A6\}) &= \{ED\} \end{aligned}$$

$$\begin{aligned} \{02\} \cdot \{87\} &= 0001\ 0101 \\ \{03\} \cdot \{6E\} &= 1011\ 0010 \\ \{46\} &= 0100\ 0110 \\ \{A6\} &= 1010\ 0110 \\ &\overline{\quad\quad\quad\quad} \\ &0100\ 0111 = \{47\} \end{aligned}$$

MIX COLOUMN TRANSFORMATION

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

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

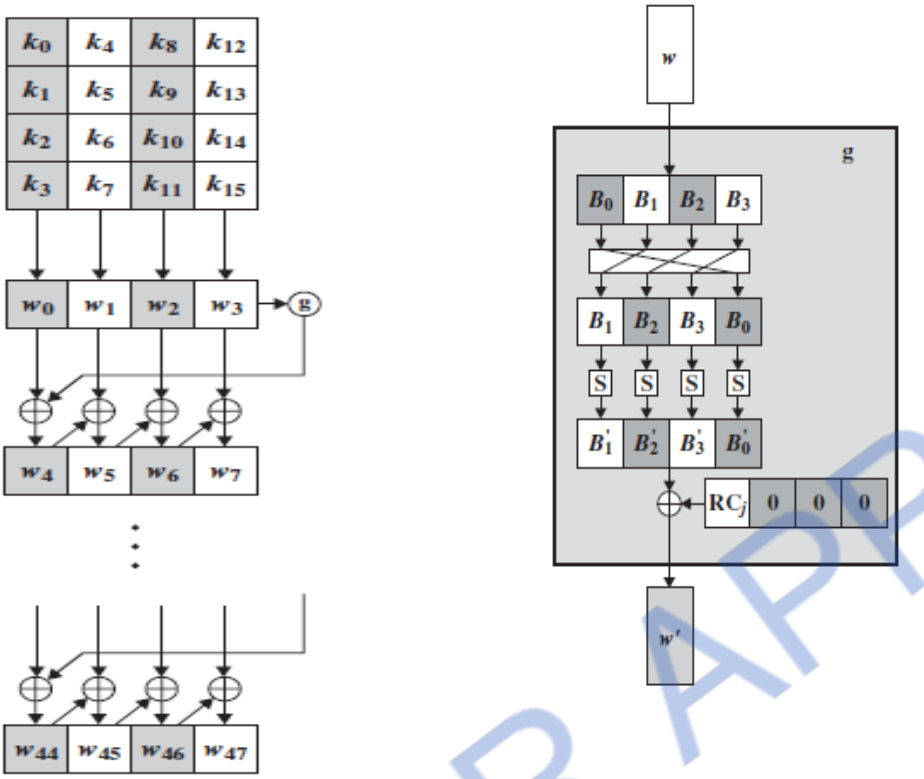
The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.

```
KeyExpansion (byte key[16], word w[44])
{
word temp
for (i = 0; i < 4; i++) w[i] = (key[4*i], key[4*i+1],
key[4*i+2],
key[4*i+3]);
for (i = 4; i < 44; i++)
{
temp = w[i – 1];
if (i mod 4 = 0) temp = SubWord XOR (RotWord (temp))
Rcon[i/4];
w[i] = w[i–4] XOR temp
}
}
```

1. RotWord performs a one-byte circular left shift on a word.This means that an input word [B0, B1, B2, B3] is transformed into [B1, B2, B3, B0].

2. SubWord performs a byte substitution on each byte of its input word, using the S-box.

3. The result of steps 1 and 2 is XORed with a round constant Rcon[j]



j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

AES KEY EXPANSION

2.16 RC4 ENCRYPTION ALGORITHM

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

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

Example:

```
RC4 Encryption
10011000 ? 01010000 = 11001000


RC4 Decryption
```

```
11001000 ? 01010000 = 10011000
```

1) Key-Generation Algorithm –

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

Key-Scheduling Algorithm:

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

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

```
for
    i = 0 to 255 do S[i] = i;
T[i] = K[i mod k - len];
```

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

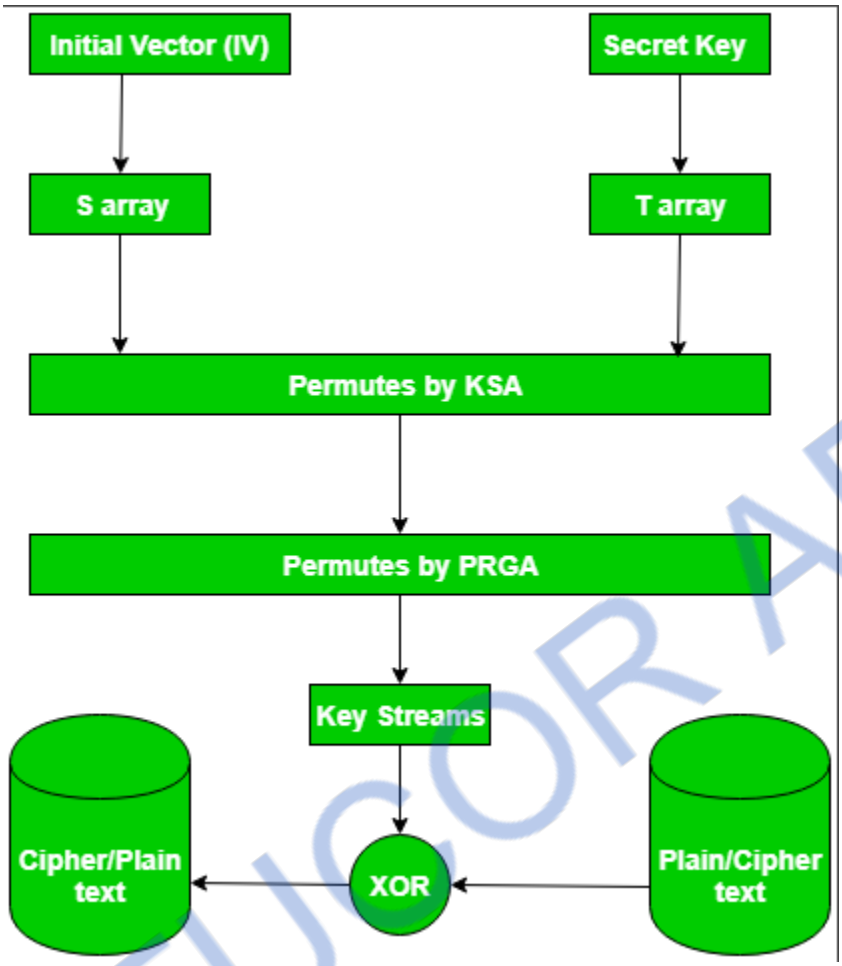
```
j = 0;
for
    i = 0 to 255 do
    {
        j = (j + S[i] + T[i])mod 256;
        Swap(S[i], S[j]);
    }
```

2) Pseudo random generation algorithm (Stream Generation):

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

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

3) Encrypt using X-Or()



RC4 Algorithm

In the RC4 encryption algorithm, the key stream is completely independent of the plaintext used. An 8 * 8 S-Box (S0 S255), where each of the entries is a permutation of the numbers 0 to 255, and the permutation is a function of the variable length key. There are two counters i, and j, both initialized to 0 used in the algorithm.

The algorithm uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream which is XORed with the plaintext to give the ciphertext. Each element in the state table is swapped at least once.

The key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits. RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure SQL.

The algorithm works in two phases, key setup and ciphering. Key setup is the first and most difficult phase of this encryption algorithm. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulas. A modulo operation is the process of yielding a remainder from division. For example, 11/4 is 2 remainder 3; therefore eleven mod four would be equal to three.

Strengths of RC4

The difficulty of knowing where any value is in the table.

The difficulty of knowing which location in the table is used to select each value in the sequence.

Encryption is about 10 times faster than DES.

Limitations of RC4

RC4 is no longer considered secure.

One in every 256 keys can be a weak key. These keys are identified by cryptanalysis that is able to find circumstances under which one of more generated bytes are strongly correlated with a few bytes of the key.

A particular RC4 Algorithm key can be used only once.

2.17 KEY DISTRIBUTION

Discussed briefly in Unit 3

CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 3 NOTES

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.

Primality

Prime Numbers

- An integer p > 1 is prime number, if its divisors are +/- 1 and +/1 p
- Any non negative integer a > 1 can be factored in the form as
- where p1 < p2 < ... < pt are prime numbers and where each a is a positive integer.
- This is known as the fundamental theorem of arithmetic
- Examples: 91 = 7 x 13, 3600 = 24 x 32 x 52

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

Miller-Rabin Algorithm

- also known as Rabin-Miller algorithm, or the Rabin-Miller test, or the Miller-Rabin test
- typically used to test a large number for primality

Two Properties of Prime Numbers

- If p is prime and a is a positive integer less than p, then a^2 mod p = 1, if and only if either a mod p = 1 or a mod p = -1 mod p = p – 1
- Let p be a prime number greater than 2. We can then write p - 1 = 2^kq with k > 0, q odd

Algorithm

```
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 mod n = 1 then return("inconclusive");
4. for j = 0 to k − 1 do
5. if a^(2^j q) mod n = n − 1 then return("inconclusive");
6. return("composite");
```

Chinese Remainder Theorem

the CRT says it is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli

Formula

$$M = \prod_{i=1}^{k} m_i$$

where the m_i are pairwise relatively prime; that is, $\gcd(m_i, m_j) = 1$ for $1 \leq i, j \leq k$, and $i \neq j$.

Relatively Prime

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

TWO ASSERTION OF CRT

1. The mapping of Equation (8.7) is a one-to-one correspondence (called a **bijection**) between Z_M and the Cartesian product $Z_{m_1} \times Z_{m_2} \times \ldots \times Z_{m_k}$. That is, for every integer A such that $0 \leq A \leq M$, there is a unique k-tuple $(a_1, a_2, \ldots, a_k)$ with $0 \leq a_i < m_i$ that represents it, and for every such k-tuple $(a_1, a_2, \ldots, a_k)$, there is a unique integer A in Z_M.
2. Operations performed on the elements of Z_M can be equivalently performed on the corresponding k-tuples by performing the operation independently in each coordinate position in the appropriate system.

The following is an example of a set of equations with different moduli:

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

The solution to this set of equations is given in the next section; for the moment, note that the answer to this set of equations is $x = 23$. This value satisfies all equations: $23 \equiv 2 \pmod 3$, $23 \equiv 3 \pmod 5$, and $23 \equiv 2 \pmod 7$.

Solution

The solution to the set of equations follows these steps:

1. Find $M = m_1 \times m_2 \times \cdots \times m_k$. This is the common modulus.
2. Find $M_1 = M/m_1$, $M_2 = M/m_2$, ..., $M_k = M/m_k$.
3. Find the multiplicative inverse of $M_1, M_2, \ldots, M_k$ using the corresponding moduli (m_1, m_2, ..., m_k). Call the inverses $M_1^{-1}, M_2^{-1}, \ldots, M_k^{-1}$.
4. The solution to the simultaneous equations is

$$x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \cdots + a_k \times M_k \times M_k^{-1}) \bmod M$$

Note that the set of equations can have a solution even if the moduli are not relatively prime but meet other conditions. However, in cryptography, we are only interested in solving equations with coprime moduli.

From the previous example, we already know that the answer is $x = 23$. We follow the four steps.

1. $M = 3 \times 5 \times 7 = 105$
2. $M_1 = 105 / 3 = 35$, $M_2 = 105 / 5 = 21$, $M_3 = 105 / 7 = 15$
3. The inverses are $M_1^{-1} = 2$, $M_2^{-1} = 1$, $M_3^{-1} = 1$
4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

Prime Factorisation

- to factor a number n is to write it as a product of other numbers: $n=a \times b \times c$
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the prime factorisation of a number n is when its written as a product of primes
  - eg. $91=7\times13$; $3600=2^4\times3^2\times5^2$

$$a = \prod_{p \in P} p^{a_p}$$

Relatively Prime Numbers & GCD

- two numbers a, b are relatively prime if have no common divisors apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300=2^1\times3^1\times5^2$ $18=2^1\times3^2$ hence $GCD(18,300)=2^1\times3^1\times5^0=6$

Fermat's Theorem

- $a^{p-1} \bmod p = 1$
  - where p is prime and gcd(a,p)=1
- also known as Fermat's Little Theorem
- useful in public key and primality testing
- Example a=7 and p=19

Euler Totient Function ø(n)

- when doing arithmetic modulo n
- complete set of residues is: 0..n-1
- reduced set of residues is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
- number of elements in reduced set of residues is called the Euler Totient Function ø(n)
- to compute ø(n) need to count number of elements to be excluded
- in general need prime factorization, but
  - for p (p prime) ø(p) = p-1
  - for p,q (p,q prime) ø(p,q) = (p-1)(q-1)
- eg.
  - ø(37) = 36

ø(21) = (3–1)×(7–1) = 2×6 = 12

Example

Prime Number

An integer $p > 1$ is a prime number if and only if its only divisors are ± 1 and $\pm p$.

Fermat's Theorem

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

Example:

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

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

Euler's Totient Function

The number of positive integers less than n and relatively prime to n. ø(1)=1

$$ø(p) = p - 1$$

suppose that we have two prime numbers p and q with $p \neq q$. Then we can show that, for $n = pq$,
$ø(n) = ø(pq) = ø(p) * ø(q) = (p - 1) * (q - 1)$

Example:

$ø(21) = ø(3) * ø(7) = (3 - 1) * (7 - 1) = 2 * 6 = 12$
where the 12 integers are {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}.

Euler's Theorem

Euler's theorem states that for every a and n that are relatively prime:
$a^{ø(n)} \equiv 1 \pmod{n}$

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

Testing for Primality
Used to determine whether a given number is prime number or not.

Miller-Rabin Algorithm
any positive odd integer $n \geq 3$ can be expressed as
$n - 1 = 2^k q$ with $k > 0$, q odd
$n - 1$ is an even integer. Then, divide $(n - 1)$ by 2 until the result is an odd number q, for a total of k divisions. If n is expressed as a binary number, then the result is achieved by shifting the number to the right until the rightmost digit is a 1, for a total of k shifts.

Properties of Prime Numbers:
First Property:
If p is prime and a is a positive integer less than p, then $a^2 \bmod p = 1$ if and only if either $a \bmod p = 1$ or $a \bmod p = -1 \bmod p = p - 1$.
By the rules of modular arithmetic $(a \bmod p)\ (a \bmod p) = a^2 \bmod p$.
Thus, if either $a \bmod p = 1$ or $a \bmod p = -1$, then $a^2 \bmod p = 1$.
Conversely, if $a^2 \bmod p = 1$, then $(a \bmod p)^2 = 1$, which is true only for
$a \bmod p = 1$ or $a \bmod p = -1$.

Second Property:
Let p be a prime number greater than 2.
$n - 1 = 2^k q$ with $k > 0$, q odd

Let a be any integer in the range $1 < a < p - 1$. Then one of the two following conditions is true.

1. a^q is congruent to 1 modulo p. That is, $a^q \bmod p = 1$, or equivalently, $a^q = 1 \pmod{p}$.
2. One of the numbers $a^q, a^{2q}, a^{4q}, \ldots\ldots, a^{2^{k-1}q}$ is congruent to -1 modulo p. That is, there is some number j in the range $(1 \leq j \leq k)$ such that $a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$ or equivalently, $a^{2^{j-1}q} = -1 \pmod{p}$.

3.5 DISCRETE LOGARITHMS

More generally, we can say that 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 a is a primitive root of n, then its powers

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

are distinct (mod n) and are all relatively prime to n. In particular, for a prime number p, if a is a primitive root of p, then

$$a, a^2, \ldots, a^{p-1}$$

are distinct (mod p). For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.

Logarithms for Modular Arithmetic

With ordinary positive real numbers, the logarithm function is the inverse of exponentiation. An analogous function exists for modular arithmetic.

Let us briefly review the properties of ordinary logarithms. The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number. That is, for base x and for a value y,

$$y = x^{\log_x(y)}$$

The properties of logarithms include

$$\log_x(1) = 0$$

$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z) \quad \textbf{(8.11)}$$

$$\log_x(y^r) = r \times \log_x(y) \quad \textbf{(8.12)}$$

Consider a primitive root a for some prime number p (the argument can be developed for nonprimes as well). Then we know that the powers of a from 1 through $(p - 1)$ produce each integer from 1 through $(p - 1)$ exactly once. We also know that any integer b satisfies

$$b = r \pmod p \quad \text{for some } r, \text{ where } 0 \le r \le (p - 1)$$

by the definition of modular arithmetic. It follows that for any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b = a^i \pmod p \quad \text{where } 0 \le i \le (p - 1)$$

This exponent i is referred to as the **discrete logarithm** of the number b for the base $a \pmod p$. We denote this value as $\text{dlog}_{a,p}(b)$.[10]

Note the following:

$$\text{dlog}_{a,p}(1) = 0 \quad \text{because } a^0 \bmod p = 1 \bmod p = 1 \quad \textbf{(8.13)}$$

$$\text{dlog}_{a,p}(a) = 1 \quad \text{because } a^1 \bmod p = a \quad \textbf{(8.14)}$$

Here is an example using a nonprime modulus, $n = 9$. Here $\phi(n) = 6$ and $a = 2$ is a primitive root. We compute the various powers of a and find

$$2^0 = 1 \qquad 2^4 = 7 \pmod 9$$

$$2^1 = 2 \qquad 2^5 = 5 \pmod 9$$

$$2^2 = 4 \qquad 2^6 = 1 \pmod 9$$

$$2^3 = 8$$

This gives us the following table of the numbers with given discrete logarithms (mod 9) for the root $a = 2$:

Logarithm	0	1	2	3	4	5
Number	1	2	4	8	7	5

To make it easy to obtain the discrete logarithms of a given number, we rearrange the table:

Number	1	2	4	5	7	8
Logarithm	0	1	2	5	4	3

Now consider

$$x = a^{\text{dlog}_{a,p}(x)} \bmod p \qquad y = a^{\text{dlog}_{a,p}(y)} \bmod p$$

$$xy = a^{\text{dlog}_{a,p}(xy)} \bmod p$$

Using the rules of modular multiplication,

$$xy \bmod p = [(x \bmod p)(y \bmod p)] \bmod p$$

$$a^{\text{dlog}_{a,p}(xy)} \bmod p = \left[\left(a^{\text{dlog}_{a,p}(x)} \bmod p\right)\left(a^{\text{dlog}_{a,p}(y)} \bmod p\right)\right] \bmod p$$

$$= \left(a^{\text{dlog}_{a,p}(x)+\text{dlog}_{a,p}(y)}\right) \bmod p$$

But now consider Euler's theorem, which states that, for every a and n that are relatively prime,

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

Any positive integer z can be expressed in the form $z = q + k\phi(n)$, with $0 \le q < \phi(n)$. Therefore, by Euler's theorem,

$$a^z = a^q \pmod n \qquad \text{if } z = q \bmod \phi(n)$$

Applying this to the foregoing equality, we have

$$\text{dlog}_{a,p}(xy) = [\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)] (\text{mod}\,\phi(p))$$

and generalizing,

$$\text{dlog}_{a,p}(y^r) = [r \times \text{dlog}_{a,p}(y)] (\text{mod}\,\phi(p))$$

This demonstrates the analogy between true logarithms and discrete logarithms.

Table 8.4 Tables of Discrete Logarithms, Modulo 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

3.6 ASYMMETRIC KEY CIPHERS:

RSA cryptosystem

The best known and widely regarded as most practical public-key scheme was proposed by

Rivest, Shamir & Adleman in 1977:

It is a public-key scheme which may be used for encrypting messages, exchanging keys, and

creating digital signatures

RSA is a public key encryption algorithm based on exponentiation using modular arithmetic to use the scheme, first generate keys:

Key-Generation by each user consists of:

- selecting two large primes at random (~100 digit), p, q
- calculating the system modulus R=p.q p, q primes
- selecting at random the encryption key e,
- e < R, gcd(e, F(R)) = 1
- solving the congruence to find the decryption key d,
- e.d [[equivalence]] 1 mod [[phi]](R) 0 <= d <= R
- publishing the public encryption key: K1={e,R}
- securing the private decryption key: K2={d,p,q}

Encryption of a message M to obtain ciphertext C is:

C = Me mod R 0 <= d <= R

Decryption of a ciphertext C to recover the message M is:

M = Cd = Me.d = M1+n.[[phi]](R) = M mod R

The RSA Algorithm

Key Generation

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

Encryption

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

Decryption

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

Example:
P=17 q=11 e=7 M=88

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17 * 11 = 187.
3. Calculate ø(n) = (p - 1)(q - 1) = 16 * 10 = 160.
4. Select e such that e is relatively prime to ø(n) = 160 and less than ø(n); we choose e = 7.
5. Determine d such that de ≡ 1 (mod 160) and d <160. The correct value is d = 23, because 23 * 7 = 161 = (1 * 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}.
Plaintext input of M = 88.

For encryption,
calculate $C = 88^7 \bmod 187$.
$88^7 \bmod 187 = [(88^4 \bmod 187) * (88^2 \bmod 187) * (88^1 \bmod 187)] \bmod 187$
$88^1 \bmod 187 = 88$
$88^2 \bmod 187 = 7744 \bmod 187 = 77$
$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$
$88^7 \bmod 187 = (88 * 77 * 132) \bmod 187 = 894{,}432 \bmod 187 = 11$

For decryption,
calculate $M = 11^{23} \bmod 187$
$11^{23} \bmod 187 = [(11^1 \bmod 187) * (11^2 \bmod 187) * (11^4 \bmod 187) * (11^8 \bmod 187) * (11^8 \bmod 187)] \bmod 187$
$11^1 \bmod 187 = 11$
$11^2 \bmod 187 = 121$
$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$
$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$
$11^{23} \bmod 187 = (11 * 121 * 55 * 33 * 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$

The Security of RSA

Brute force: This involves trying all possible private keys.
Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.
Timing attacks: These depend on the running time of the decryption algorithm.
Hardware fault-based attack: This involves inducing hardware faults in the processor that is generating digital signatures.
Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

Solve:
a. p = 3; q = 11, e = 7; M = 5
b. p = 5; q = 11, e = 3; M = 9
c. p = 7; q = 11, e = 17; M = 8

d. p = 11; q = 13, e = 11; M = 7

3.7 KEY MANAGEMET AND KEY DISTRIBUTION

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others.

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

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

A Key Distribution Scenario

User A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, K_a, known only to itself and the KDC; similarly, B shares the
master key K_b with the KDC. The following steps occur.

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using K_a. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

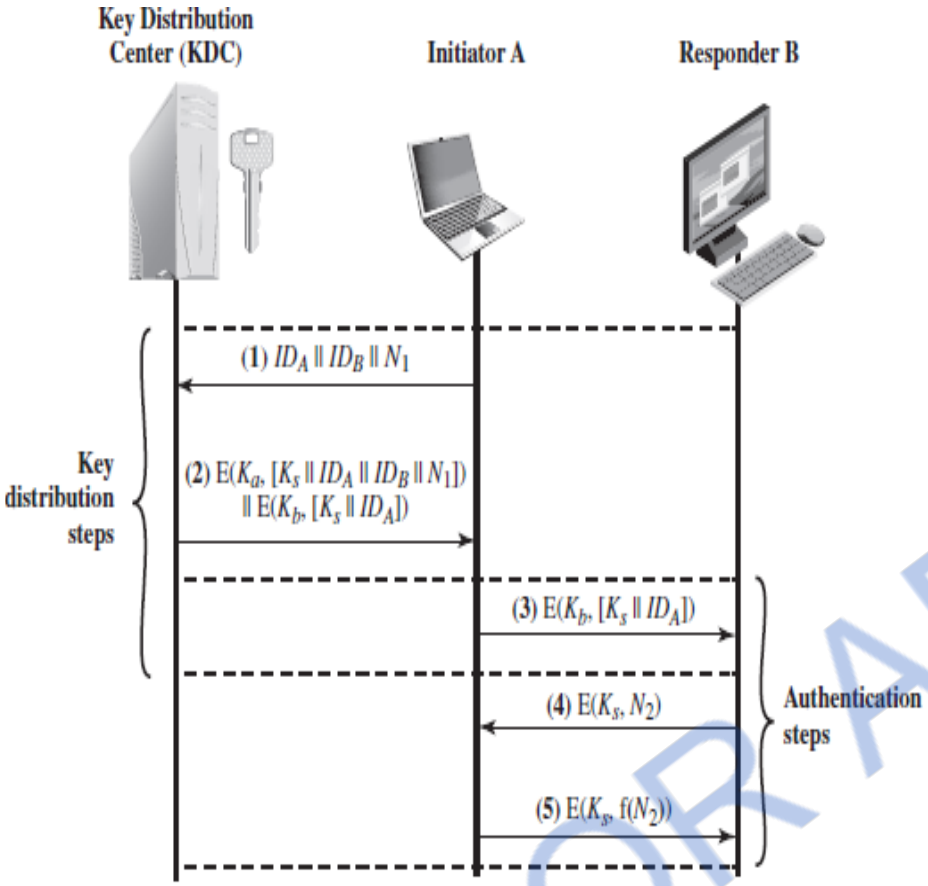
• The one-time session key, K_s, to be used for the session
• The original request message, including the nonce, to enable A to match

this response with the appropriate request Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

In addition, the message includes two items intended for B:

• The one-time session key, K_s, to be used for the session
• An identifier of A (e.g., its network address), ID_A

These last two items are encrypted with K_b (the master key that the KDC
shares with B). They are to be sent to B to establish the connection and prove A's identity.



3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b,[K_s \| ID_A])$.

Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).
4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.
5. Also, using Ks, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2

Hierarchical Key Control

It is not necessary to limit the key distribution function to a single KDC.
A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities.

Session Key Lifetime

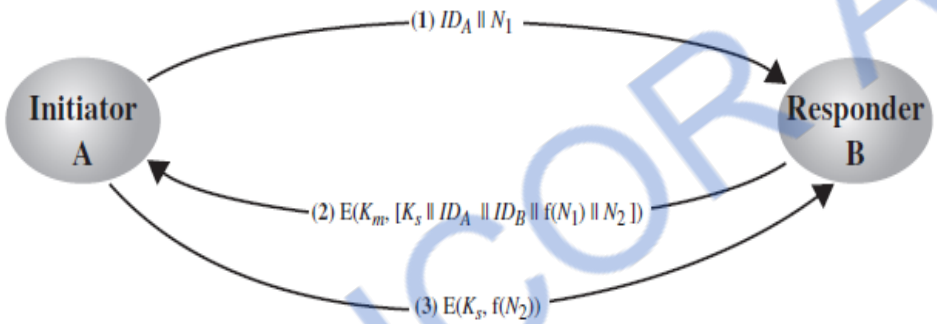
The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

For connection-oriented protocols, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session. If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.

For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination. Thus, it is not obvious how often one needs to change the session key. The most secure approach is to use a new session key for each exchange. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction. A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

Decentralized Key Control

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.
A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution. Thus, there may need to be as many as [n(n - 1)]/2 master keys for a configuration with n end systems.



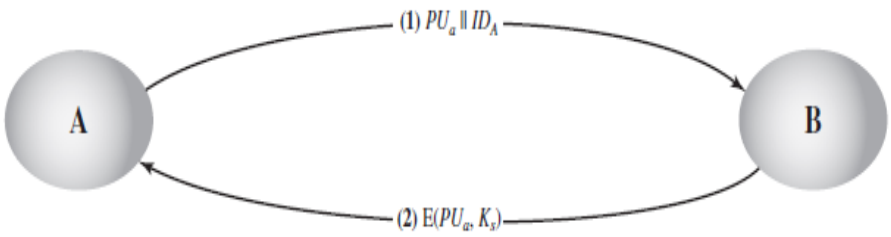
A session key may be established with the following sequence of steps

1. A issues a request to B for a session key and includes a nonce, N_1.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$, and another nonce, N_2.
3. Using the new session key, A returns $f(N_2)$ to B.

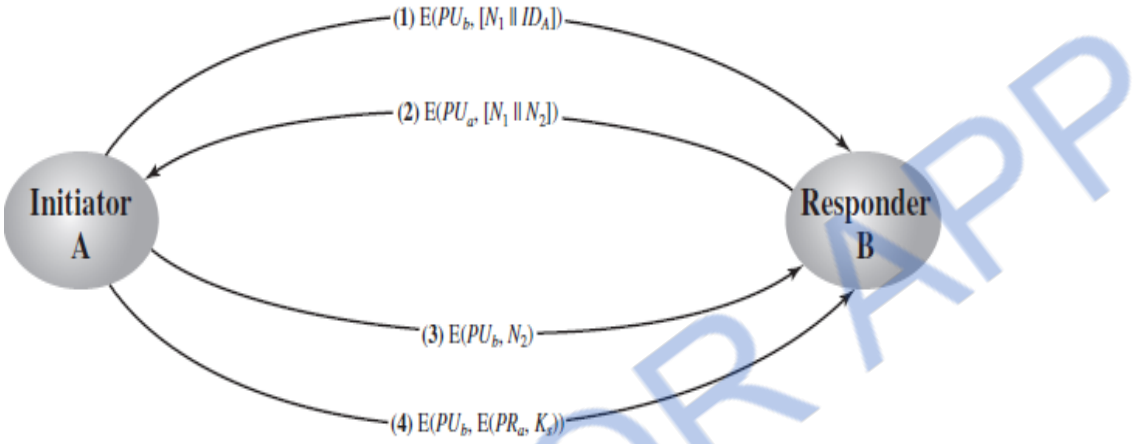
Symmetric Key Distribution using Asymmetric Encryption

Simple Secret Key Distribution

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A.
2. B generates a secret key, K_s, and transmits it to A, which is encrypted with A's public key.
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s.
4. A discards PU_a and PR_a and B discards PU_a.



Public-Key Distribution of Secret Keys



1. A uses B's public key to encrypt a message to B containing an identifier of A(ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends M = E(PU_b, E(PR_a, K_s)) to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes D(PU_a, D(PR_b, M)) to recover the secret key.

Distribution of Public Keys

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

Public Announcement of Public Keys

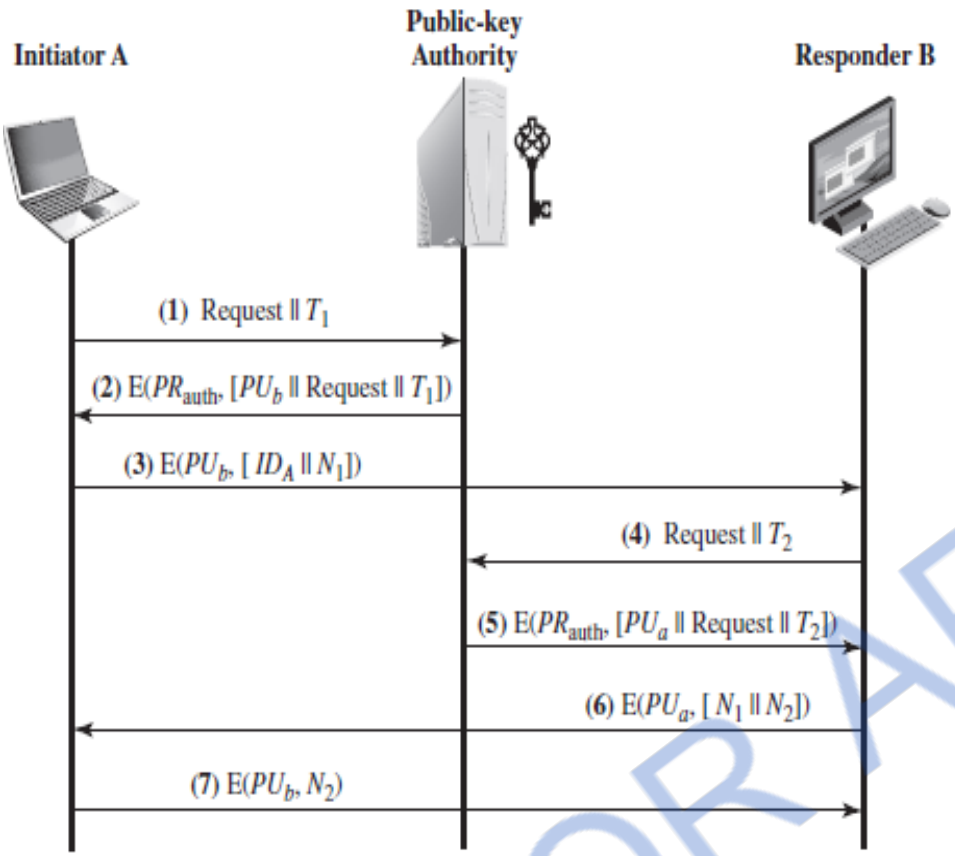
Publicly Available Directory

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

Public-Key Authority

1. A sends a timestamped message to the public-key authority containing a
Request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth}. Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

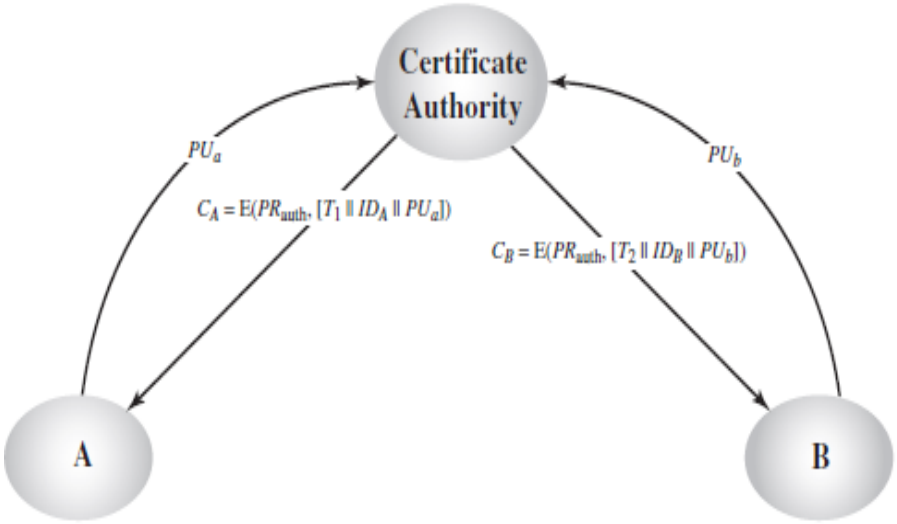
- B's public key, PU_b, which A can use to encrypt messages destined for B
- The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not
altered before reception by the authority
- The original timestamp given so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



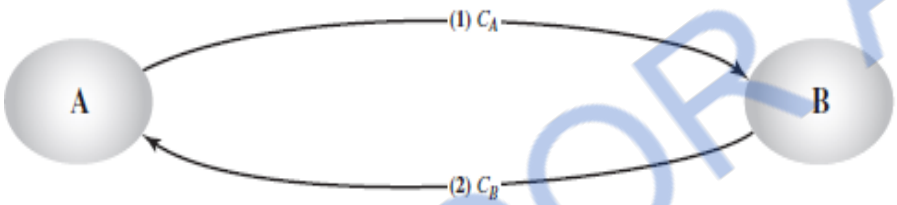
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
4, 5. B retrieves A's public key from the authority in the same manner as A
Retrieved B's public key.
At this point, public keys have been securely delivered to A and B, and they
may begin their protected exchange. However, two additional steps are desirable:
6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.
7. A returns N_2, which is encrypted using B's public key, to assure B that its
Correspondent is A.

Public-Key Certificates

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



(a) Obtaining certificates from CA



(b) Exchanging certificates

For participant A, the authority provides a certificate of the form
$C_A = E(PR_{auth}, [T \| ID_A \| PU_a])$

where PR_{auth} is the private key used by the authority and T is a timestamp.
$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T \| ID_A \| PU_a])) = (T \| ID_A \| PU_a)$

3.8 DIFFIE HELLMAN KEY EXCHANGE

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

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

Diffie Hellman key exchange as follows:

There are publicly known numbers: a prime number „q" and an integer α that is primitive root of q.

suppose users A and B wish to exchange a key. User A selects a random integer XA < q and computes

YA = α XA mod q. Similarly, user B independently selects a random

integer XB < q and computes YB = α XB mod q. Each side keeps the X value private and

makes the Y value available publicly to the other side. User A computes the key as

K = (YB)XA mod q and

User B computes the key as

K = (YA)XB mod q

These two calculations produce identical results.

K = (YB)XA mod q

= (α XB mod q)XA mod q

= (α XB)XA mod q

= (α XA)XB mod q

= (α XA mod q)XB mod q

= (YA)XB mod q

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

The security of the algorithm lies in the fact that, while it is relatively easy to calculate Exponentials

modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Diffie-Hellman Key Exchange

Global Public Elements

q	prime number
α	α<q, and α is a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

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

Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

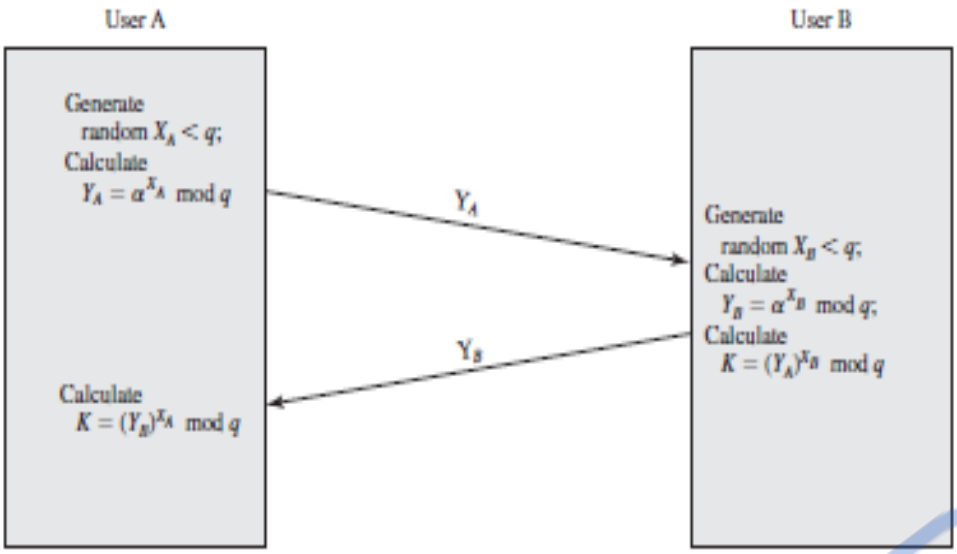
Example:

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 private keys $X_A = 97$ and $X_B = 233$, respectively.

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

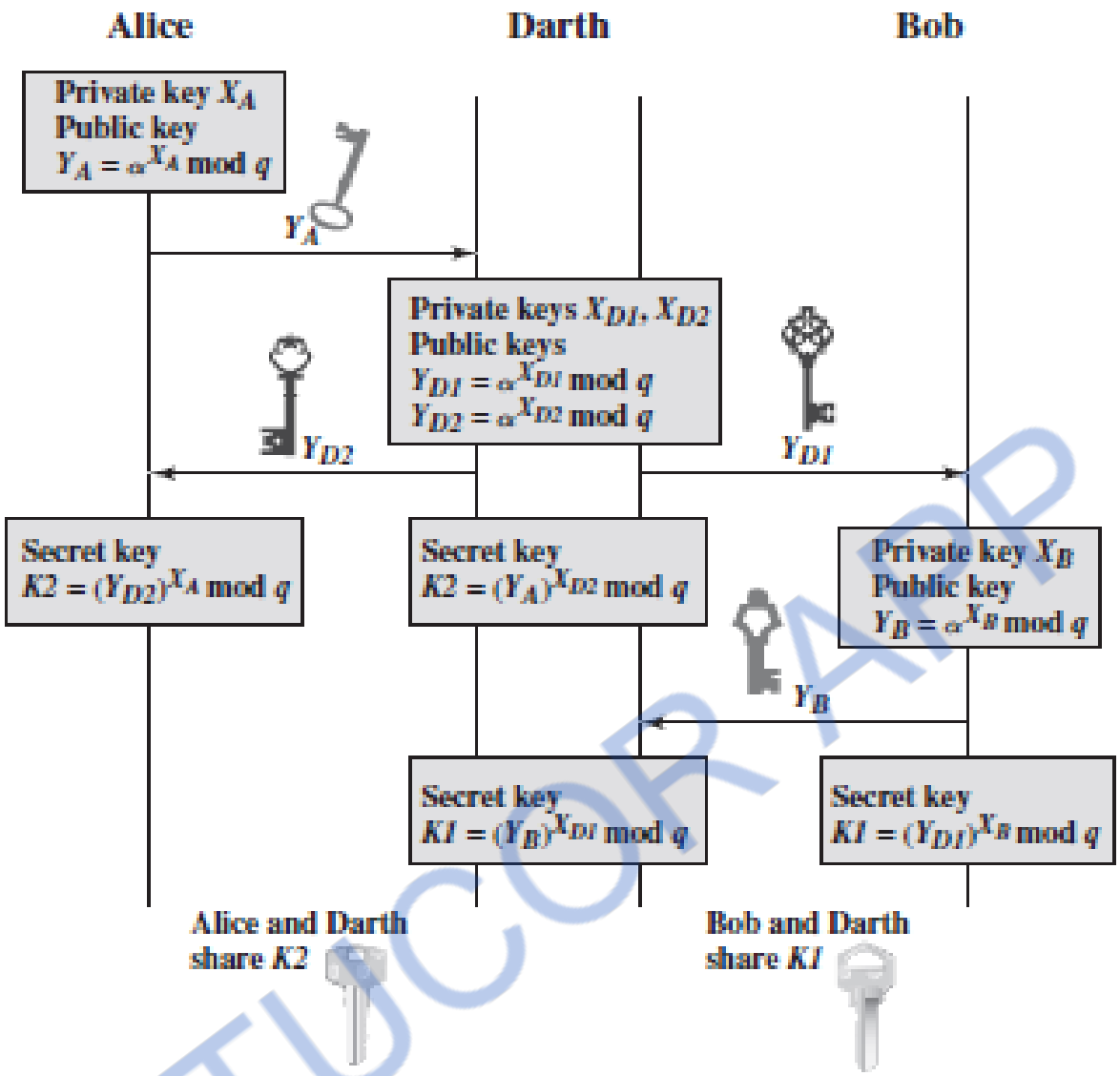
Key Exchange Protocols



Man-in-the-Middle Attack

Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows.

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates $K_2 = (Y_A)^{XD2} \bmod q$.
4. Bob receives Y_{D1} and calculates $K_1 = (Y_{D1})^{XB} \bmod q$.
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates $K_1 = (Y_B)^{XD1} \bmod q$.
7. Alice receives Y_{D2} and calculates $K_2 = (Y_{D2})^{XA} \bmod q$.



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. Alice sends an encrypted message M: $E(K_2, M)$.
2. Darth intercepts the encrypted message and decrypts it to recover M.
3. Darth sends Bob $E(K_1, M)$ or $E(K_1, M')$, where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

3.9 The Elgamal Cryptosystem

The variant of the Diffie-Hellman key distribution scheme, allowing secure exchange of

Messages published in 1985 by ElGamal in T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms",

Like Diffie-Hellman its security depends on the difficulty of factoring logarithms

Key Generation

- select a large prime p (~200 digit), and
- [[alpha]] a primitive element mod p
- A has a secret number xA
- B has a secret number xB
- A and B compute yA and yB respectively, which are then made public

yA = [[alpha]]xA mod p
yB = [[alpha]]xB mod p

to **encrypt** a message **M** into ciphertext **C**,

- selects a random number **k,** 0 <= k <= p-1
- computes the message key **K**

K = yB
k mod p

- computes the ciphertext pair: C = {c1,c2}

C1 = [[alpha]]k mod p C2 = K.M mod p

to **decrypt** the message

- extracts the message key **K**

K = C1
xB mod p = [[alpha]]k.xB mod p

- extracts **M** by solving for M in the following equation:

C2 = K.M mod p

Algorithm

Global Public Elements

q	prime number
α	α<q, and α is a primitive root of q

Key Generation by Alice

Select private X_A	$X_A < q - 1$

Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$\{q, a, Y_A\}$
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	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2K^{-1}) \bmod q$

1. Bob generates a random integer k.
2. Bob generates a one-time key K using Alice's public-key components Y_A, q, and k.
3. Bob encrypts k using the public-key component a, yielding C_1. C_1 provides sufficient information for Alice to recover K.
4. Bob encrypts the plaintext message M using K.
5. Alice recovers K from C1 using her private key.
6. Alice uses K^{-1} to recover the plaintext message from C_2.

Thus, K functions as a one-time key, used to encrypt and decrypt the message.

Example:

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

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

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. $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = KM \bmod q = 7 * 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{XA} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2K^{-1}) \bmod q = 5 * 11 \bmod 19 = 55 \bmod 19 = 17$.

If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block. If k is used for more than one block, knowledge of one block M1 of the message enables the user to compute other blocks as follows.
Let
$C_{1,1} = \alpha^k \bmod q$; $C_{2,1} = KM_1 \bmod q$
$C_{1,2} = \alpha^k \bmod q$; $C_{2,2} = KM_2 \bmod q$
Then,
$C_{2,1}/C_{2,2} = (KM_1 \bmod q / KM_2 \bmod q) = (M_1 \bmod q / M_2 \bmod q)$
If M_1 is known, then M_2 is easily computed as
$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$

3.10 Elliptic Curve Cryptography (ECC)

The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead.

Abelian Group:
an abelian 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

(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′ in G such that a • a′ = a′ • a = e.
(A5) Commutative: a•b = b•a for all a, b in G.

Elliptic Curves over Real Numbers

Elliptic curves are not ellipses. They are so named because they are described by cubic equations, similar to those used for calculating the circumference of an ellipse. In general, cubic equations for elliptic curves take the following form, known as a **Weierstrass equation:**

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, e are real numbers and x and y take on values in the real numbers.

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

Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also included in the definition of an elliptic curve is a single element denoted O and called the point at infinity or the zero point

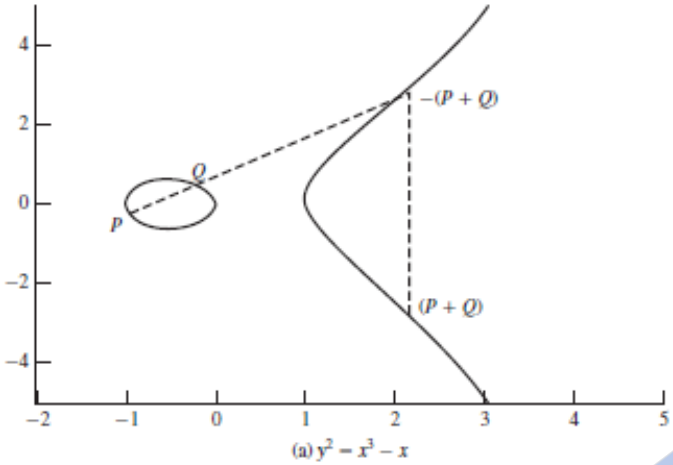
$$y = \sqrt{x^3+ax+b}$$

1. O serves as the additive identity. Thus O = -O; for any point P on the elliptic curve, P + O = P. In what follows, we assume $P \neq O$ and $Q \neq O$.
2. The negative of a point P is the point with the same x coordinate but the negative of the y coordinate; that is, if P = (x, y), then -P = (x, -y).
Note that these two points can be joined by a vertical line.
Note that P + (-P) = P - P = O.
3. To add two points P and Q with different x coordinates, draw a straight line between them and find the third point of intersection R. It is easily seen that there is a unique point R that is the point of intersection

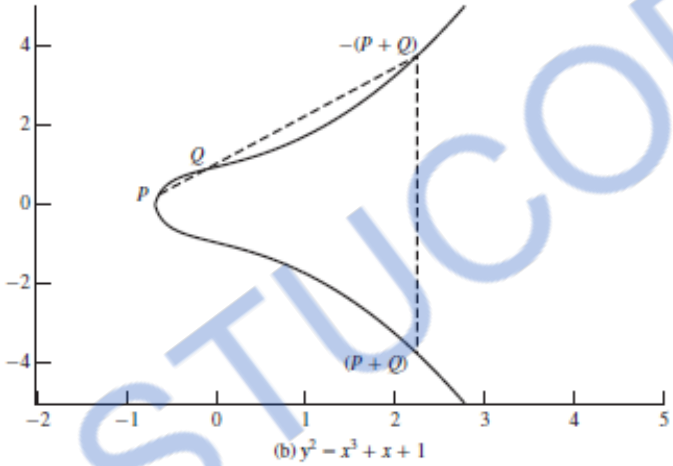
(unless the line is tangent to the curve at either P or Q, in which case we take R = P or R = Q, respectively). To form a group structure, we need to define addition on these three points: P + Q = -R. That is, we define P + Q to be the mirror image (with respect to the x axis) of the third point of intersection.

4. The geometric interpretation of the preceding item also applies to two points, P and -P, with the same x coordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point. We therefore have P + (-P) = O, which is consistent with item (2).

5. To double a point Q, draw the tangent line and find the other point of intersection S. Then Q + Q = 2Q = -S.



(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

Algebraic Description of Addition

$P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, that are not negatives of each other, the slope of the line l that joins them is $\Delta = (y_Q - y_P)/(x_Q - x_P)$. There is exactly one other point where l intersects the elliptic curve, and that is the negative of the sum of P and Q.

After some algebraic manipulation, we can express the sum R = P + Q as

$x_R = \Delta^2 - x_P - x_Q$

$y_R = -y_P + \Delta(x_P - x_R)$

P + P = 2P = R. When $y_P \neq 0$

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P$$

$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P$$

ECC Diffie-Hellman Key Exchange

Global Public Elements

Eq(a, b)	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A * G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B * G$

Calculation of Secret Key by User A

$K = n_A * P_B$

Calculation of Secret Key by User B

$K = n_B * P_A$

CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 4 NOTES

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

4.1AUTHENTICATION REQUIREMENT

Communication across the network, the following attacks can be identified.

Disclosure – release of message contents to any person or process not possessing the appropriate cryptographic key.

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.

Masquerade – insertion of messages into the network from fraudulent source. This can be creation of message by the attacker using the authorized port.

Content modification – changes to the contents of a message, including insertion, deletion, transposition, and modification.

Sequence modification – any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

Timing modification – delay or replay of messages.

- In a connection oriented application, an entire session or sequence of messages could be replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.
- In a connectionless application, an individual message could be delayed or replayed.

Source repudiation – denial of transmission of message by source.

Destination repudiation – denial of receipt of message by destination.

4.2AUTHENTICATION FUNCTION

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels.

At the lower level, there must be some sort of function that produces an authenticator, a value to be used to authenticate a message.

At the higher-level, low-level function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

The types of function that may be used to produce an authenticator are grouped into three classes.

Message Encryption – the ciphertext of the entire message serves as its authenticator.

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.

Hash Function – a public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator.

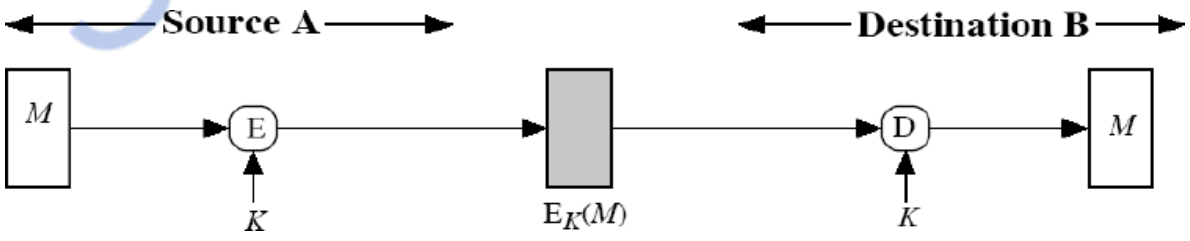
Message Encryption:

Message encryption Message encryption by itself can provide a measure of authentication.

The analysis differs from symmetric and public key encryption schemes.

(a) If symmetric encryption is used then:

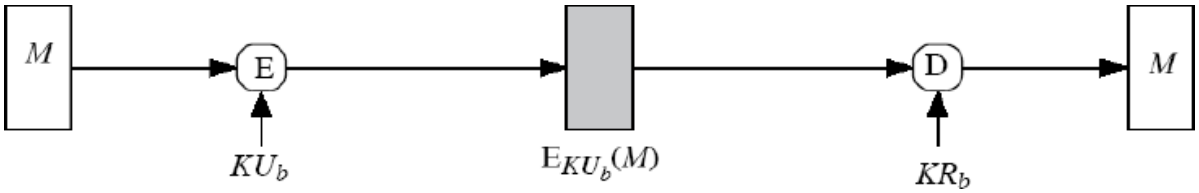
- A message m, transmitted from source A to destination B is encrypted using a secret key shared by A and B.
- Since only sender and receiver knows key used
- Receiver knows sender must have created it. Hence authentication is provided.
- Know content cannot have been altered. Hence confidentiality is also provided.
- If message has suitable structure, redundancy or a checksum to detect any changes
- Therefore Symmetric Encryption provides authentication and confidentiality.



Symmetric key encryption confidentiality, authentication and signature

(b) If public-key encryption is used:

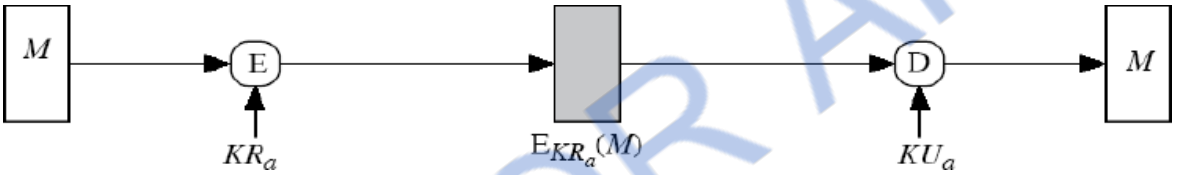
This method is the use of public key cryptography which provides confidentiality only. The sender A makes use of the public key of the receiver to encrypt the message. Here there is no authentication because any user can use B‟s public key to send a message and claim that only A has sent it.



Public key encryption confidentiality

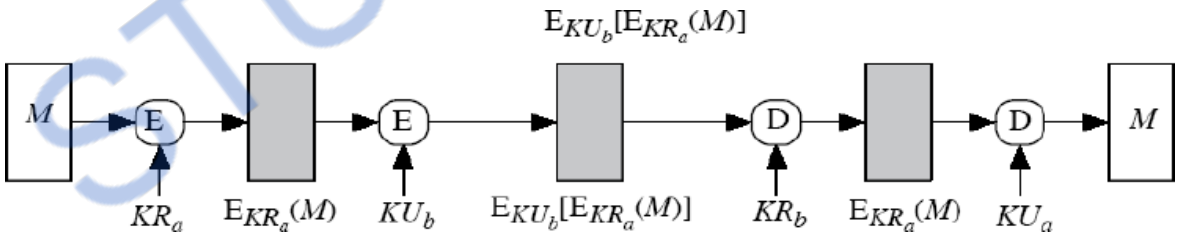
In this method to have only authentication, the message is encrypted with the sender"s A"s private key. The receiver B uses the sender"s A"s public key to decrypt the message. Now A cannot deny that it has not transmitted since it only knows its private key. This is called as authentication or Digital Signature. Hence the problem is the,

- Receiver cannot determine whether the packet decrypted contains some useful message or random bits.
- The problem is that anyone can decrypt the message when they know the public key of sender A.



Public key encryption authentication and signature

This method provides authentication, confidentiality and digital signature. But the problem with this method is the complex public key cryptography algorithm should be applied twice during encryption and twice during decryption.



Public key encryption confidentiality, authentication and signature

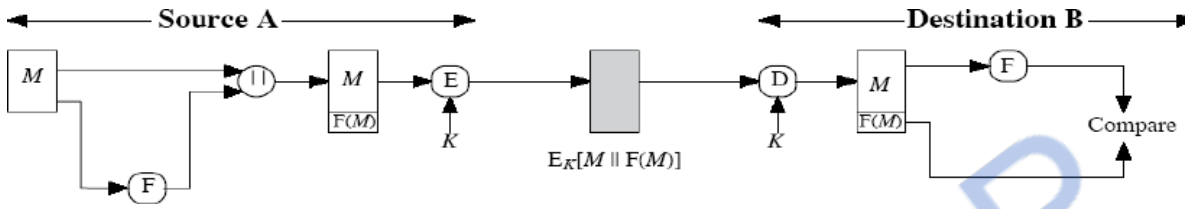
Suppose the message can be any arbitrary bit pattern, in that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message.

One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function.

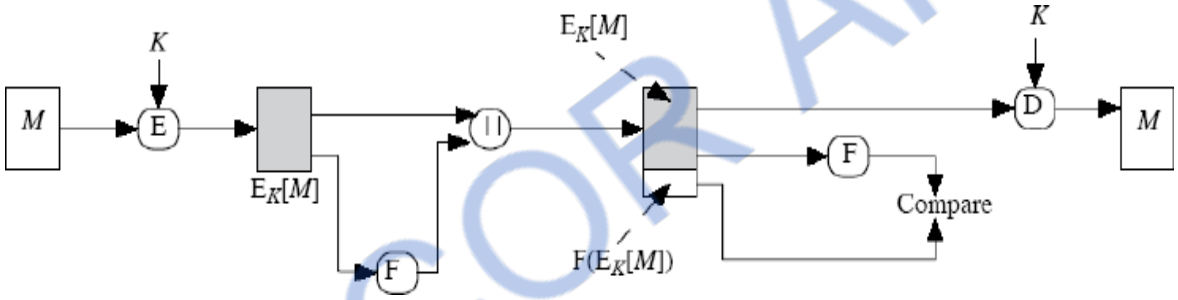
Append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption „A" prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted.

At the destination, B decrypts the incoming block and treats the result as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS.

If the calculated FCS is equal to the incoming FCS, then the message is considered authentic. In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext.



Internal error control



External error control

4.3 MAC

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message.

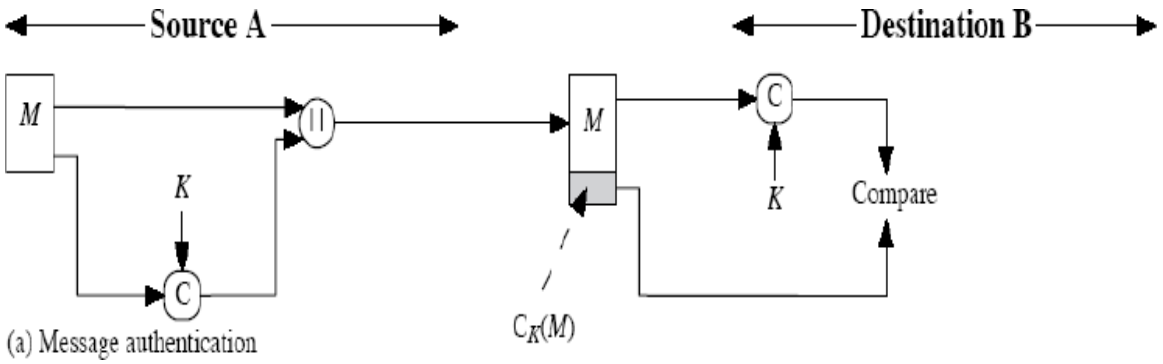
This technique assumes that two communication parties say A and B, share a common secret key „k". When A has to send a message to B, it calculates the MAC as a function of the message and the key.

$$MAC = CK (M)$$

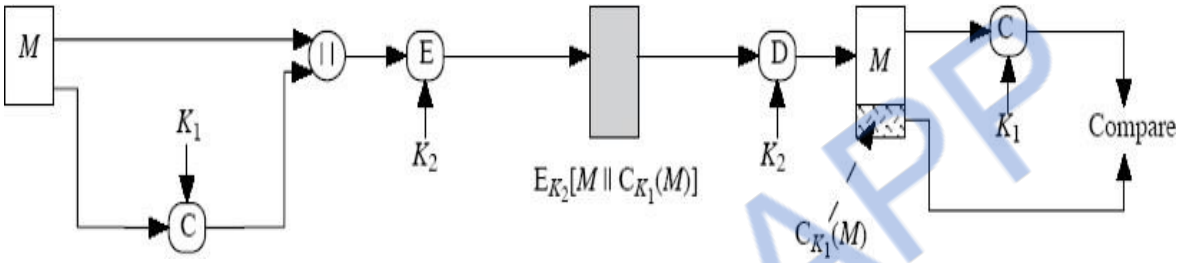
Where M – input message C – MAC function K – Shared secret key The

message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC.

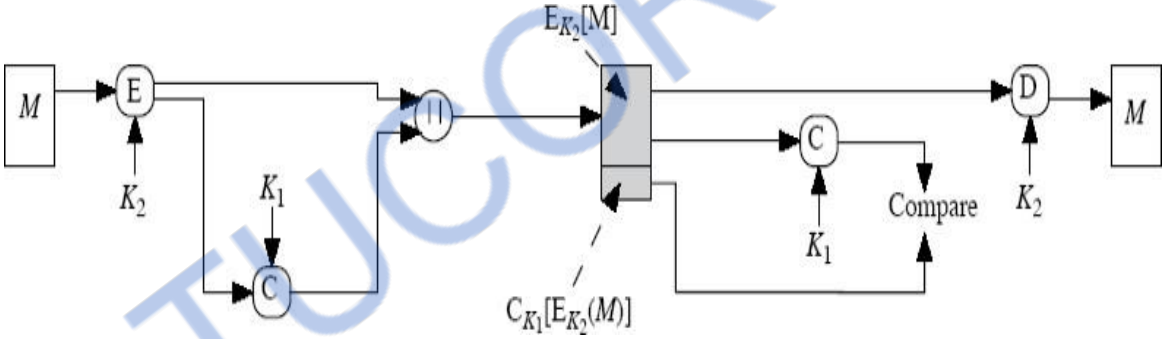
The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic. A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function.



Message Authentication



Message authentication and confidentiality, authentication tied to plain text



Message authentication and confidentiality, authentication tied to ciphertext Requirements for MAC:

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key.

Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require 2(k-1) attempts for a k-bit key.

If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the

MAC size. Then, given a known M_1 and MAC_1, with $MAC_1 = C_K(M_1)$, the cryptanalyst can perform $MAC_i = CK_i(M1)$ for all possible key values Ki.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.

Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

Round 1

Given: M_1, $MAC_1 = C_K(M_1)$
Compute $MAC_i = C_{Ki}(M_1)$ for all 2^k keys Number of matches $\approx 2^{(k-n)}$

Round 2

Given: M_2, $MAC_2 = C_K(M_2)$
Compute $MAC_i = C_{Ki}(M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1Number of matches $\approx 2^{(k-2xn)}$ and so on

Consider the following MAC algorithm. Let $M = (X_1||X_2||...||X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i. Then define

$$\Delta(M) = X_1 + X_2 \ldots X_m$$
$$C_k(M) = E_k(\Delta(M))$$

Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 2^{56} encryptions.

But the opponent can attack the system by replacing X1 through X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows:

$$Y_m = Y_1 + Y_2 \ldots\ldots Y_{m1} + \Delta(M)$$

The opponent can now concatenate the new message, which consists of Y_1 through Y_m, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 $X_{(m-1)}$ bits can be fraudulently inserted.

Then the MAC function should satisfy the following requirements: The MAC function should have the following properties:

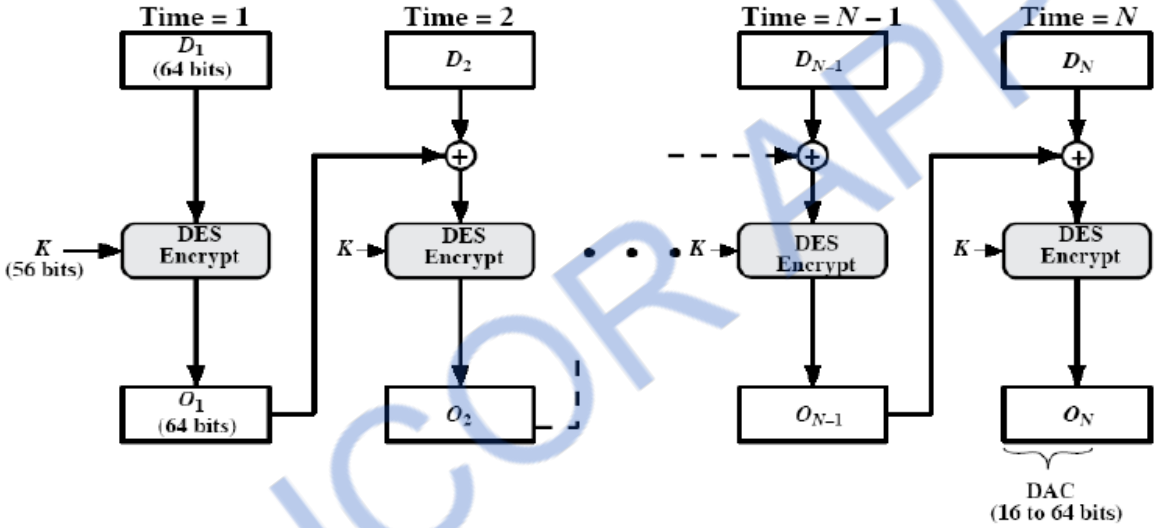
- If an opponent observes M and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message M‟ such that $C_K(M') = C_K(M)$
- $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M‟, the probability that $C_K(M) = C_K(M')$ is 2^{-n} where n is the number of bits in the MAC.
- Let M‟ be equal to some known transformation on M. i.e., M‟ = f(M).

MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES.

The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: D_1, D_2 … D_n. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code (DAC) is calculated as follows:

$O_1 = E_K(D_1)$
$O_2 = E_K(D_2 + O_1)$ O_3
$= E_K(D_3 + O_2)$
……
$O_N = E_K(D_N + O_{N-1})$



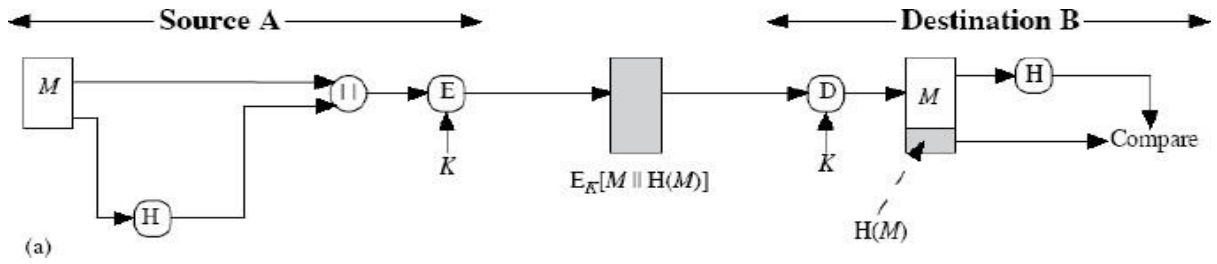
Data Authentication Algorithm

4.4 HASH FUNCTION

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

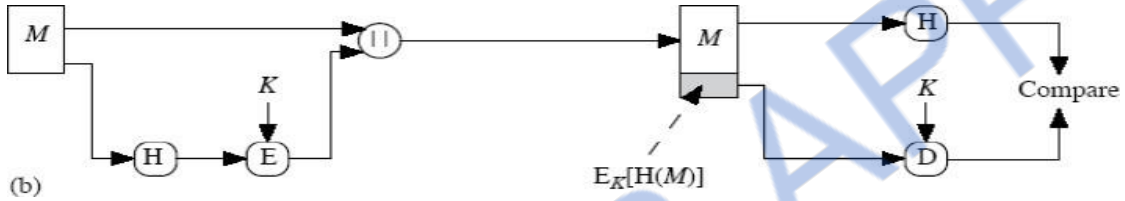
Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value. There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the hash code, confidentiality is also provided.

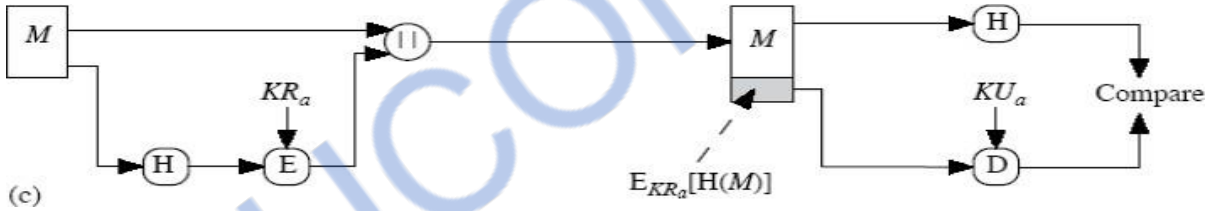


Hash Function

Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

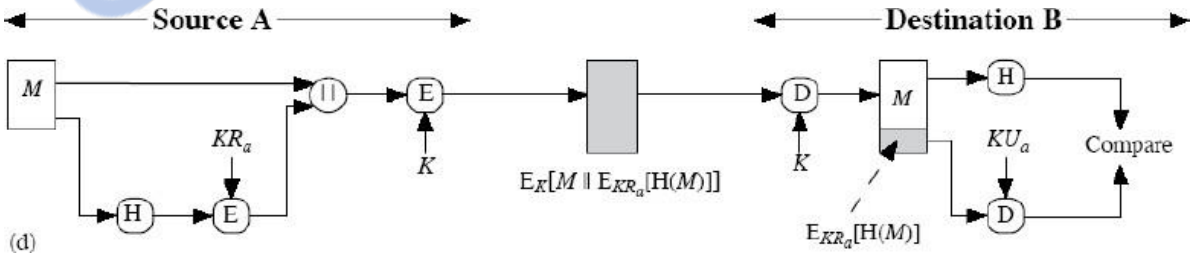


Only the hash code is encrypted, using the public key encryption and using the sender"s private key. It provides authentication plus the digital signature.

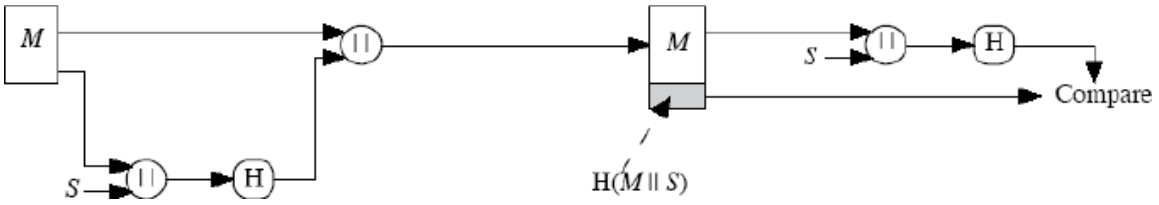


Basic use of Hash Function

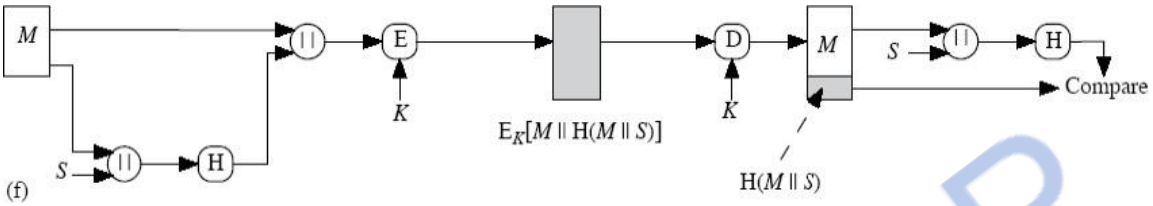
If confidentiality as well as digital signature is desired, then the message plus the public key encrypted hash code can be encrypted using a symmetric secret key.



This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value „S". The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.



Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.



Basic use of Hash Function

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

$$h = H(M)$$

where M is a variable-length message and H(M) is the fixed-length hash value.

The hash value 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 hash value.

Requirements for a Hash Function

1. H can be applied to a block of data of any size.

2. H produces a fixed-length output.

3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.

5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as weak collision resistance.

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as strong collision resistance.

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code.

The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used.

The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) 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. One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.

This can be expressed as follows: $C_i = b_{i1} + b_{i1} \ldots + b_{im}$

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} = ith bit in jth block

Procedure:

1. Initially set the n-bit hash value to zero.

2. Process each successive n-bit block of data as follows:

   a. Rotate the current hash value to the left by one bit.
   b. XOR the block into the hash value.

Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted message M, then an opponent would need to find an M' such that H(M') = H(M) to substitute another message and fool the receiver.

On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message.

However, a different sort of attack is possible, based on the birthday paradox. 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.

1. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. (Fraudulent message)

2. 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.

3. 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^{32}

MEET-IN-THE-MIDDLE ATTACK.

Divide a message M into fixed-size blocks $M_1, M_2, ..., M_N$ and use a symmetric encryption system such as DES to compute the hash code G as follows:

H_o = initial value H_i = EMi [Hi-1]

$G = H_N$

This is similar to the CBC technique, but in this case there is no secret key. As with any hash code, this scheme is subject to the birthday attack and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.

Here is the scenario; we assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long:

1. Calculate the unencrypted hash code G.

2. Construct any desired message in the form $Q_1, Q_2, ..., Q_{N2}$.

3. Compute for $H_i = EQ_i [H_{i-1}]$ for $1 \leq i \leq (N-2)$.

4. Generate $2^{m/2}$ random blocks; for each block X, compute $E_X[H_{N-2}.]$ Generate an additional $2^{m/2}$ random blocks; for each block Y, compute $D_Y[G]$, where D is the decryption function corresponding to E.

5. Based on the birthday paradox, with high probability there will be an X and Y such that $E_X [H_{N-2}] = D_Y[G]$.

6. Form the message $Q_1, Q_2,..., Q_{N-2}$, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

4.5 SECURITY OF HASH FUNCTION AND MAC

Just as with symmetric and public-key encryption, 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.

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm.

Requirements of Hash Function:

One-way: For any given code h, it is computationally infeasible to find x such that H(x) = h.

Weak collision resistance: For any given block x, it is computationally infeasible to find y x with H(y) = H(x).

Strong collision resistance: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

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	$2^{n/2}$

Message Authentication Codes

A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.. To attack a hash code, we can proceed in the following way. 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.

To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

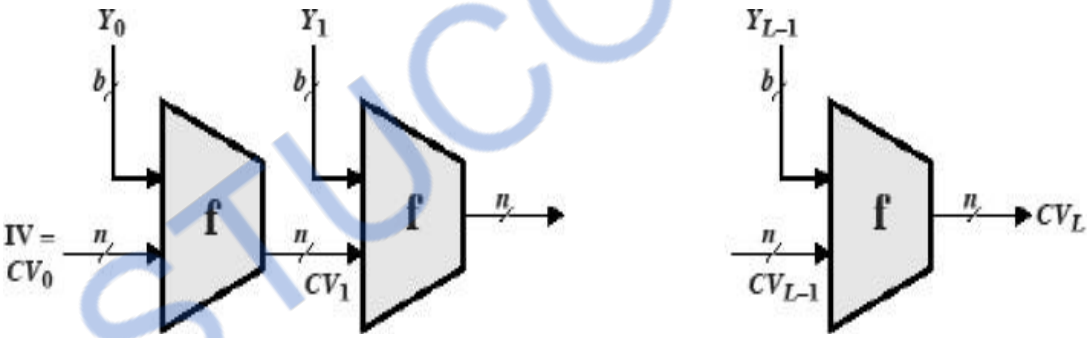
Cryptanalysis

As with encryption algorithms, 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.

Hash Functions

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits. The final block also includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult.

Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.



IV=Initial Value
Y_i = ith input block
n=Length of Hash code

CV=Changing Variable
L=number of input blocks
b=Length of input block

General structure of secure hash code

The hash algorithm involves repeated use of a compression function,f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) 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:

CV0 = IV = initial n-bit value CVI = f(CVi-1, Yi-1) 1 ≤i ≤L H(M) = CVL

Where the input to the hash function is a message M consisting of the blocks Yo, Y1,..., YL-1. The structure can be used to produce a secure hash function to operate on a message of any length.

Message Authentication Codes :

There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs.

4.6 SHA

The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

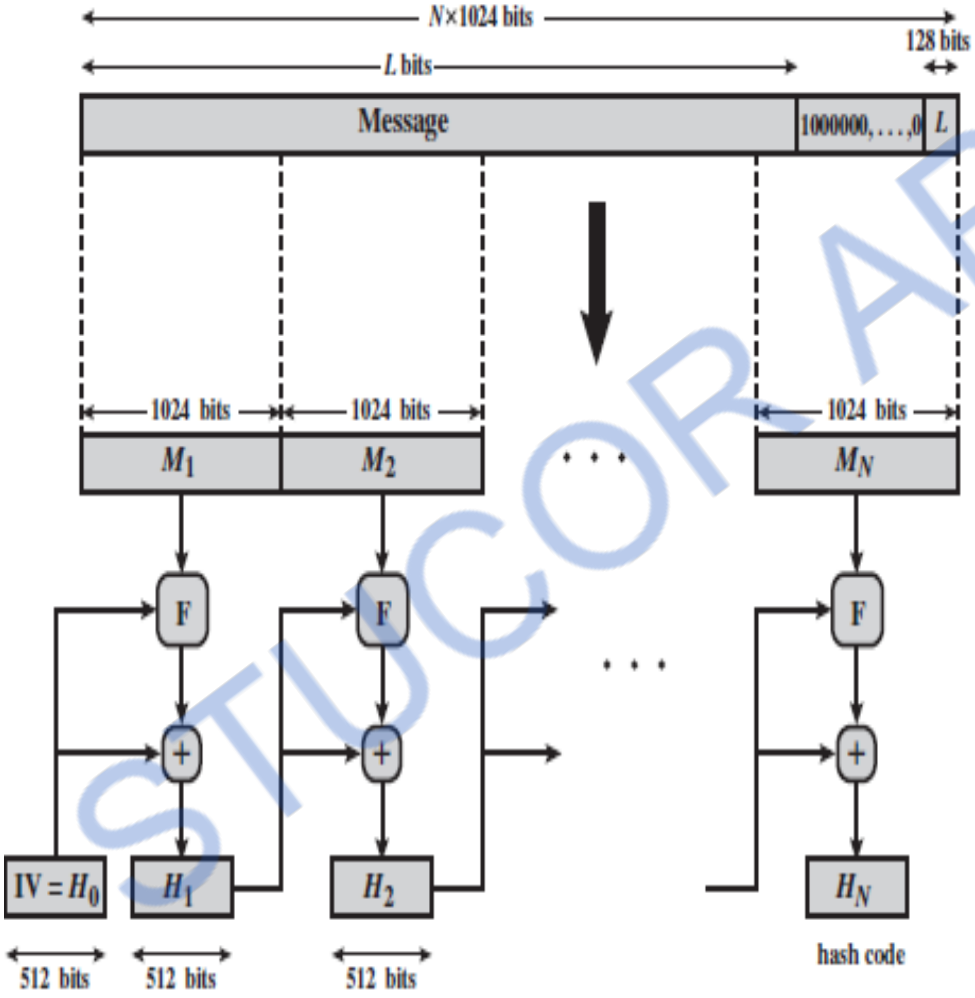
Comparison of SHA Parameters

	SHA I	SHA 224	SHA 256	SHA 384	SHA 512
Message Digest Size	160	224	256	384	512
Message Size	<264	<264	<264	<2128	<2128
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Step 1
Append padding bits.

The message is padded so that its length is congruent to 896 modulo 1024 [length K 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step 2
Append length.
A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).



Message Digest Generation Using SHA-512

Step 3
Initialize hash buffer.
A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).

a = 6A09E667F3BCC908
b = BB67AE8584CAA73B
c = 3C6EF372FE94F82B
d = A54FF53A5F1D36F1
e = 510E527FADE682D1
f = 9B05688C2B3E6C1F
g = 1F83D9ABFB41BD6B
h = 5BE0CD19137E2179

Step 4

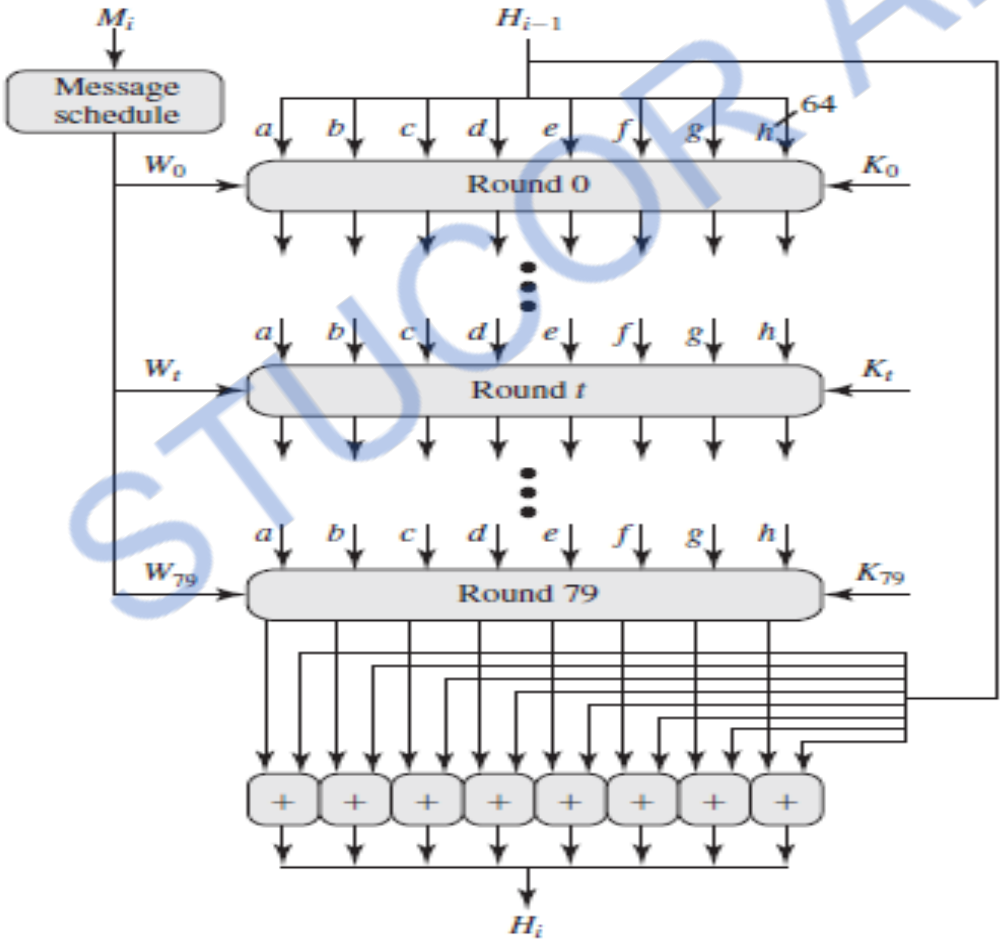
Process message in 1024-bit (128-word) blocks.

The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F.

Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, Hi-1. Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi). These values are derived using a message schedule described subsequently.

Each round also makes use of an additive constant Kt, where $0 \leq t \leq 79$ indicates one of the 80 rounds.

The output of the eightieth round is added to the input to the first round (Hi-1) to produce Hi. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in Hi-1, using addition modulo 264.

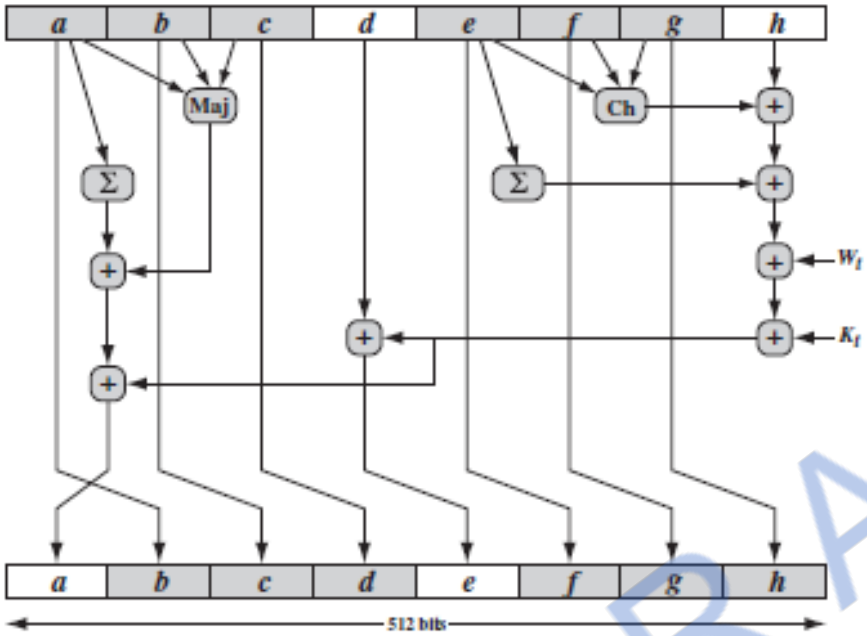


SHA-512 Processing of a Single 1024-Bit Block

Step 5

Output.

After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.



Elementary SHA Operation (single step)

Summary of SHA-512

H0 = IV
Hi = SUM64(Hi-1, abcdefghi)
MD = HN

where,
IV = initial value of the abcdefgh buffer, defined in step 3
abcdefghi = the output of the last round of processing of the ith message block
N = the number of blocks in the message (including padding and length fields)
SUM64 = addition modulo 264 performed separately on each word of the pair of inputs
MD = final message digest value

SHA-512 Round Function

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$

h = g
g = f
f = e
e = d + T1
d = c
c = b
b = a
a = T1 + T2

t = step number; $0 \le t \le 79$

$\text{Ch}(e, f, g)$ = $(e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$
the conditional function: If e then f else g

$\text{Maj}(a, b, c)$ = $(a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

$\left(\sum_0^{512} a\right)$ = $\text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right)$ = $\text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

Wt = a 64-bit word derived from the current 1024-bit input block
Kt = a 64-bit additive constant
+ = addition modulo 264

4.7 DIGITAL SIGNATURE AND AUTHENTICATION PROTOCOLS

Digital Signature Requirements

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other.

Disputes created by message authentication are:

- Creation of fraud message.
- Deny the sending of message

For example, suppose that John sends an authenticated message to Mary, the following disputes that could arise:

1 Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

Properties of digital signature :

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

Requirements for a digital signature:

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender, to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- 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.
- It must be practical to retain a copy of the digital signature in storage.

Direct Digital Signature

The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption). Note that it is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature.

If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

The validity of the scheme just described depends on the security of the sender‟s private key.

Weakness of Direct Digital Signature:

- If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.
- Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X"s signature and stamped with a time before or equal to T.

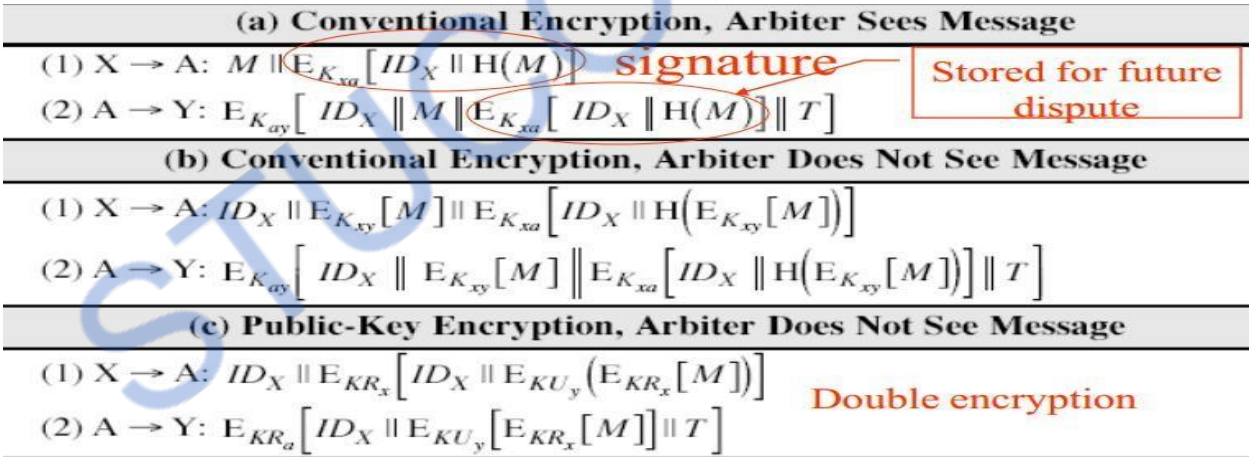
Arbitrated Digital Signatures

The problem associated with the Direct digital signature can be overcome by using arbitrated schemes.

In the arbitrated scheme, the entire signed message from the sender goes to the arbiter A. The arbiter subjects the message and signature to a number of tests to check the origin and control. The date and time is attached to the message. This indicates that the digital signature has been verified and is satisfied. The message is then transmitted to the receiver.

Requirement of the arbiter:

- As the arbiter plays a sensitive and crucial role, it should be a trusted third party.



Notation:X = Sender Y = Recipient A = Arbiter M=Message T=Timestamp

Scheme 1: Conventional encryption, Arbiter sees the message:

The sender X and arbiter A share the master key K_{ax} the receiver y and the arbiter A share the master key K_{ay}

When X wants to send a message M to Y, construct a message computes the hash value H(M). This hash is encrypted using symmetric encryption with the key Kax which acts as signature. The message along with the signature is transmitted to A.

At A, it decrypts the signature and checks the hash value to validate the message. A transmit the message to Y, encrypted with K_{ay}. Y decrypt to extract the message and signature.

Disadvantage:

Eaves dropper can read the message as there is no confidentiality.

Scheme 2: Conventional encryption, Arbiter does not see the message:

- K_{ax} and K_{ay} are the master keys.
- K_{xy} is the key shared between the X and Y
- When x wants to transmit a message to Y, the packet goes to arbiter.
- The same procedure as that of I scheme is used X transmit an identifier, a copy of the message encrypted with K_{xy} and a signature to A.
- The signature is the hash of the message encrypted with K_{xa}
- A decrypt the signature, and checks the hash value to validate the message.
- A cannot read the message, A attaches to it the time stamps, encrypt with K_{xa} and transmit to Y.

Attack: The arbiter can join with an attacker and deny a message with sender‟s signature.

Scheme 3: Public key encryption, Arbiter does not see the message:

This method uses the public key cryptography which gives authentication and digital signature.

The doubly encrypted message is concatenated with ID_x and sent to arbiter.

- A can decrypt the outer encryption to ensure that the message has come from X.
- A then transmit the message with ID_x and time

stamp. Advantages:

- No information is shared among parties before communication, hence fraud is avoided.
- No incorrectly dated message can be

sent. Disadvantages:

The complex public key algorithm is to be twice for encryption and twice for decryption.

Authentication Protocols

Authentication Protocols used to convince parties of each other‟s identity and to exchange

session keys.

Mutual Authentication

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

Key issues are

- **confidentiality** – to protect session keys and prevent masqueraded and compromised
- **timeliness** – to prevent replay attacks

Replay Attacks

Where a valid signed message is copied and later resent

- Simple replay
  The opponent simply copies the message and replays it later.
- Repetition that can be logged
  The opponent replay a time stamped message within a valid time window.
- Repetition that cannot be detected
  The attacker would have suppressed the original message from the receiver. Only the replay message alone arrives.
- Backward replay without modification
  This replay back to the sender itself.This is possible if the sender cannot easily recognize the difference between the message sent and the message received based on the content.

Countermeasures include

One approach to coping with replay attacks is to attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order.

The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with. Because of this overhead, sequence numbers are generally not used for authentication and key exchange. Instead, one of the following two general approaches is used:

- **Timestamps:** Party A accepts a message as fresh only if the message contains a **Timestamp** that is close enough to A‟s knowledge of current time. This approach requires that clocks among the various participants be synchronized.

- **Challenge/response:** Party A, expecting a fresh message from B, first sends B a **nonce** (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

Using Symmetric Encryption

- Use a two-level hierarchy of keys
- Usually with a trusted Key Distribution Center (KDC)
  - Each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - Master keys used to distribute the session ke

Needham-Schroeder Protocol

- Original third-party key distribution protocol
- For session between A and B mediated by KDC
- Protocol overview is:

1. A→KDC: $ID_A \| ID_B \| N_1$
2. KDC→A: $EK_a[K_s \| ID_B \| N_1 \| EK_b[K_s \| ID_A]]$
3. A→B: $EK_b[K_s \| ID_A]$
4. B→A: $EK_s[N_2]$
5. A→B: $EK_s[f(N_2)]$

Step 1: A to KDC ,transmit the id of source and destination and a nonce N_1 as a request.
Step 2: A securely acquires the session key in step 2 and a packet to B encrypted with EK_b is also received from KDC.
Step 3: A transmit to B the message it got from KDC.
Step 4: As a hand shake, B encrypts a new nonce N_2 and transmit to A with K_s. Step 5: As a hand shake, A encrypt the function of N_2 with K_s
Step 4 and Step 5 as used as hand shake and prevent the reply attacks.

Attacks:

- Used to securely distribute a new session key for communications between A &B
- But is vulnerable to a replay attack if an old session key has been Compromised
  - Then message 3 can be resent convincing B that is communicating With A
- Modifications to address this require:
  - Timestamps
  - Using an extra nonce

DenningProtocol:

To overcome the above weakness by a modification to the Needham/Schroeder protocol that

includes the addition of a timestamp to steps 2 and 3.

A→KDC: $ID_A||ID_B$

KDC → A: E(K_a, [K_sIIID_B *II T II* E(K_b, [K_s *IIID*$_A$ *II T*])]) A→B: E(K_b, [K_s II ID_A *IIT*])

B→A: E(K_s,

N1) A→ B:

E(K_s, f($N1$))

T is a timestamp that assures A and B that the session key has only just been generated. Thus, both A and B know that the key distribution is a fresh exchange.A and B can verify timeliness by checking that

$$I\ Clock - T\ I < \Delta t1+\Delta t2$$

The **Denning protocol** seems to provide an increased degree of security compared to the Needham/Schroeder protocol. However, a new concern is raised: namely, that this new scheme requires reliance on clocks that are synchronized throughout the network.

Suppress-replay attacks:

The problem occurs when a sender‟s clock is ahead of the intended recipient‟s clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient‟s site. This replay could cause unexpected results.

Method to overcome:

One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC‟s clock. The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonce.

This latter alternative is not vulnerable to a suppress-replay attack, because the nonce the recipient will choose in the future are unpredictable to the sender.

An attempt is made to respond to the concerns about suppress replay attacks and at the same time fix the problems in the Needham/Schroeder protocol.

The protocol is

1. A: →B: $ID_A||N_a$
2. B: → KDC: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])$

3. KDC: → A: E(K_a, [ID_B *f f*N_a*f f*Ks *f f*T_b]) „ E(Kb , [ID_A„ Ks „ T_b]) „N_b
4. A: →B: Kb, [ID_A„ K_s„ T_b]) „ E(K_s, N_b)

1. A initiates the authentication exchange by generating a nonce,N_a , and sending that plus its identifier to B in plaintext. This nonce N_a will be returned to A in an encrypted message that includes the session key, assuring A of its timeliness.

2. B alerts the KDC that a session key is needed. Its message to the KDC includes its identifier and a nonce, . This nonce will be returned to B in an encrypted message that includes the session key, assuring B of its timeliness.B‟s message to the KDC also includes a block encrypted with the secret key shared by B and the KDC. This block is used to instruct the KDC to issue credentials to A; the block specifies the intended recipient of the credentials, a suggested expiration time for the credentials, and the nonce received from A.

3. The KDC passes on to A B‟s nonce and a block encrypted with the secret key that B shares with the KDC. The block serves as a "ticket" that can be used by A for subsequent authentications, as will be seen. The KDC also sends to A block encrypted with the secret key shared by A and the KDC. This block verifies that B has received A‟s initial message (ID_B) and that this is a timely message and not a replay (N_a), and it provides A with a session key (K_S) and the time limit on its use (T_b).

A transmits the ticket to B, together with the B‟s nonce, the latter encrypted with the session key. The ticket provides B with the secret key that is used to decrypt E(K_S,N_b) to recover the message came from A and is not a replay.

Using Public-Key Encryption

- Have a range of approaches based on the use of public-key encryption
- Need to ensure have correct public keys for other parties
- Using a central authentication server (AS)
- Various protocols exist using timestamps or nonces

Denning AS Protocol

- Denning presented the following:

1. A→AS: ID_A|| ID_B
2. AS→A: E_{KRas}[ID_A||KU_a||T] || EKRas[ID_B||KU_b||T]
3. A→B: E_{KRas}[ID_A||KU_a||T] || EKRas[ID_B||KU_b||T] ||EKU$_b$[E_{KRas}[Ks||T]]

- Note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized

clocks Another approach proposed by woo and lam makes use

of nonce.

1. A→KDC: ID_A|| ID_B
2. *KDC*→A: E_{KRauth}[ID_B II K_{Ub}]

3. a→b: $EK_{Ub}[N_a$ II $ID_A]$
4. B→KDC: ID_B II ID_A II EK_{Uauth} $[N_a]$
5. KDC→B: E_{KRauth} [IDA II KUa] II $E_{KUb}[E_{KRauth}[N_a$ II K_s II $ID_B]]$
6. B→A: $E_{KUa}[E_{KRauth}[N_a$ II K_s II $ID_B]$ II $N_b]$
7. A→B: $E_{Ks}[N_b]$

Step 1: A informs the KDC of its intention to establish a secure connection with B.

Step 2: The KDC returns to A a copy of B‟s public key certificate.

Step 3: A informs B of its desire to communicate and sends a nonce N_a.

Step 4: B asks the KDC for A‟s public key certificate and request a session key.;B includes A‟s nonce so that the KDC can stamp the session key with that nonce.The nonce is protected using the KDC‟s public key.

Step 5: The KDC returns to B a copy of A‟s public key certificate, plus the information$[N_a,K_s,ID_B]$.

Step 6: The triple $[N_a,K_s,ID_B]$, still encrypted with the KDC‟s private key,is relayed to A,together with a nonce N_b generated by B.

All the foregoing are encrypted using A‟s public key. A retrieves the session key K_s and uses it to encrypt N_b and return it to B.

Step 7: Assures B of A's Knowledge of the session key

One-Way Authentication

- Required when sender & receiver are not in communications at same time (eg. Email)
- Have header in clear so can be delivered by email system
- May want contents of body protected & sender authenticated

Using Symmetric Encryption

- can refine use of KDC but can‟t have final exchange of nonce

1. A→KDC: ID_A|| ID_B|| N_1
2. KDC→A: $EK_a[K_s$ || ID_B|| N_1|| $EK_b[K_s||ID_A]$]
3. A→B: $EK_b[K_s||ID_A]$ || EKs[M]

- Does not protect against replays
  - ➢ could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- If confidentiality is major concern, can use:

$$A \rightarrow B: EKU_b[K_s] \,||\, EK_s[M]$$

In this case, the message is encrypted with a one-time secret key. A also encrypts this one-time key with B‟s public key. Only B will be able to use the corresponding private key to recover the one-time key and then use that key to decrypt the message. This scheme is more efficient than simply encrypting the entire message with B‟s public key.

- If authentication needed use a digital signature with a digital

certificate: $A \rightarrow B: M, II\ E_{KRa}(H(M))$

This method guarantees that A cannot later deny having sent the message. However, this technique is open to another kind of fraud. Bob composes a message to his boss Alice that contains an idea that will save the company money. He appends his digital signature and sends it into the e-mail system.

Eventually, the message will get delivered to Alice‟s mailbox. But suppose that Max has heard of Bob‟s idea and gains access to the mail queue before delivery. He finds Bob‟s message, strips off his signature, appends his, and requeues the message to be delivered to Alice. Max gets credit for Bob‟s idea.

To counter such a scheme, both the message and signature can be encrypted with the recipient‟s public key:

$$A \rightarrow B: E_{KUb}, [M \,||\, E_{KRa}, H(M)]$$

The latter two schemes require that B know A‟s public key and be convinced that it is timely. An
effective way to provide this assurance is the digital certificate

$$A \rightarrow B: M \,||\, EKR_a\,[H(M)] \,||\, EKR_{as}\ II\ T||\ ID_A \,||\, KU_a]$$

In addition to the message,A sends B the signature encrypted with A‟s private key and A‟s certificate encrypted with the private key of the authentication server. The recipient of the message first uses the certificate to obtain the sender‟s public key and verify that it is authentic and then uses the public key to verify the message itself.

4.8 DSS

The DSS Approach

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.

RSA approach

The message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender‟s private key to form the signature.

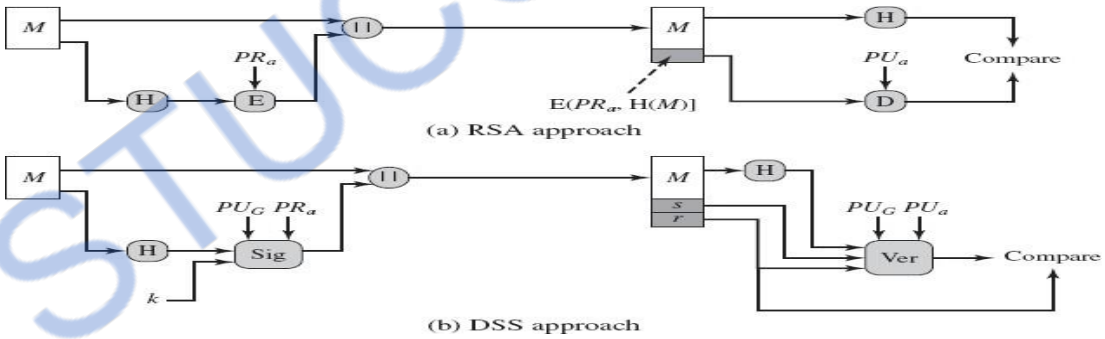
Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender‟s public key.

If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

DSS approach

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). The result is a signature consisting of two components, labeled *s* and *r*.



Two Approaches to Digital Signatures

At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender‟s public key, which is paired with the sender‟s private key.

The output of the verification function is a value that is equal to the signature component if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

The Digital Signature Algorithm:

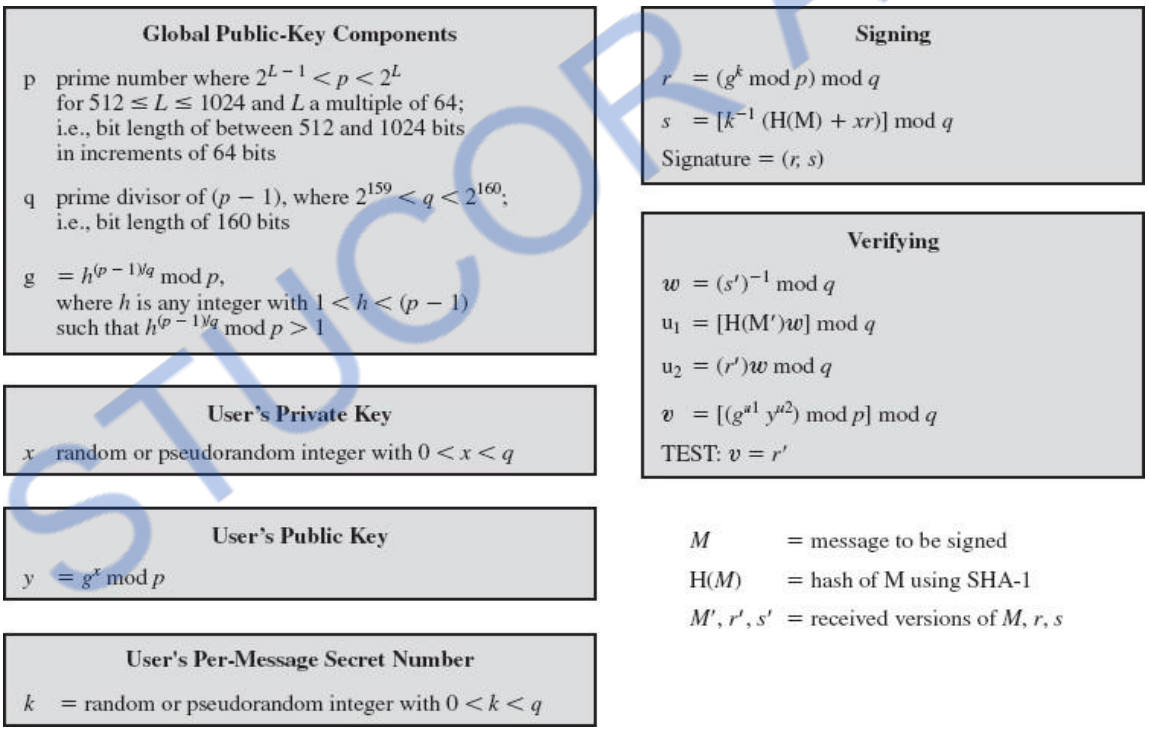
There are three parameters that are public and can be common to a group of users.

- A 160-bit prime number q is chosen.
- Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides (*p* - 1).
- Finally, *g* is chosen to be of the form $h^{(p-1)/q} \bmod p$, where *h* is an integer between 1 and (p-1).

With these numbers in hand, each user selects a private key and generates a public key.The private key x must be a number from 1 to (q-1) and should be chosen randomly. T

The public key is calculated from the private key as $y = g^x \bmod p$ .The calculation of given is relatively straightforward. However, given the public key y, it is believed to be computationally infeasible to determine x, which is the discrete logarithm of *y* to the base *g*, mod p.

At the receiving end, verification is performed using the formulas. The receiver generates a quantity v that is a function of the public key components, the sender‟s public key, and the hash code of the incoming message. If this quantity matches the component of the signature, then the signature is validated.



The Digital Signature Algorithm (DSA)

The value r does not depend on the message at all. Instead, r is a function of k and the three global public-key components.

The multiplicative inverse of k (mod q) is passed to a function that also has as inputs the message hash code and the user‟s private key.

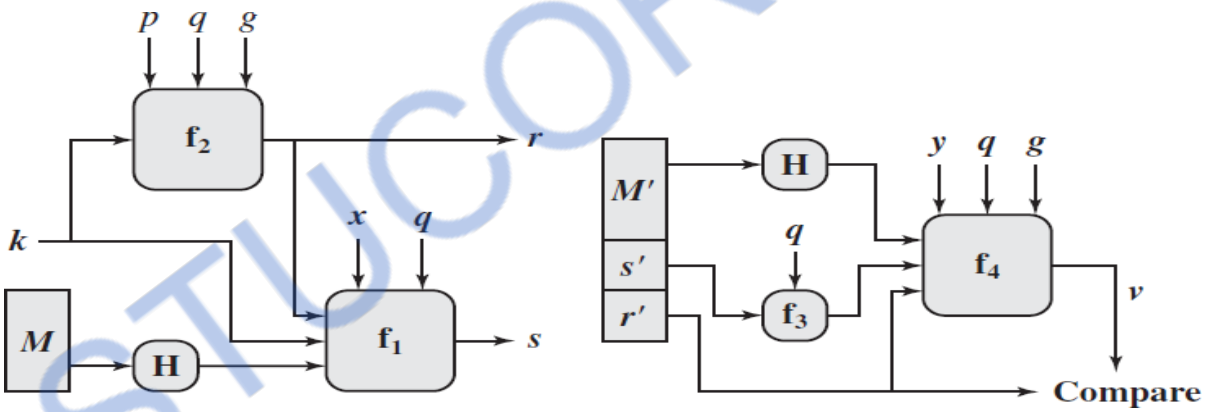
The structure of this function is such that the receiver can recover using the incoming message and signature, the public key of the user, and the global public key. Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover k from r to recover x from s.

The only computationally demanding task in signature generation is the exponential calculation $g^k \bmod p$. Because this value does not depend on the message to be signed, it can be computed ahead of time.

Selects a private key and generates a public key. The private key x must be a number from 1 to (q1) and should be chosen randomly. The public key is calculated from the private key as $y = g^x \bmod p$.

To create a signature, a user calculates two quantities, r and s, that are functions of the public key components (p, q, g), the user's private key (x), the hash code of the message, H(M), and an additional integer k that should be generated randomly and be unique for each signing.

At the receiving end, verification is performed using the formulas. The receiver generates a quantity v that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the r component of the signature, then the signature is validated.



DSS Signing and Verifying

4.9 ENTITY AUTHENTICATION

- Entity authentication is a technique designed to let one party prove the identity of another party.
- An entity can be a person, a process, a client, or a server.
- The entity whose identity needs to be proved is called the claimant;
- the party that tries to prove the identity of the claimant is called the verifier

Terminologies involved

- Verifier-The person who is in charge of checking that the correct entity is involved in communication is the Verifier. Verifier can also create some tokens by itself during communication which are used.
- Claimant-The person who wants to start communication by proving its identity is Claimant.
- Nonce–Time variant parameter which is served to distinguish one protocol instance from another like random number.
- Salt–Upon arrival of password we may add some bits upon initial entry. This t-bit random string is called “Salt”.

Entity Authentication Vs. Message Authentication

Message Authentication

This involves a meaningful message.
This doesn‟t provide timeliness guarantees as to when it was created etc.

Entity Authentication

This doesn‟t involve meaningful message; it involves some “claim” for proving its entity.
Time is important, as in this protocol corroboration of a claimant‟s identity takes place.

Data-Origin Versus Entity Authentication

There are two differences between message authentication (data-origin authentication), and entity authentication.

1)Message authentication might not happen in real time; entity authentication does.

2)Message authentication simply authenticates one message; the process needs to be repeated for each new message. Entity authentication authenticates the claimant for the entire duration of a session.

Verification Categories

Something known - They include standard password, PIN –personal identification numbers,

etc ▪

Something possessed - This includes some physical accessory like our own student id to access labs, ATM cards etc.
Something inherent - They include characteristics like finger prints, handwritten signatures, voice, i.e. some human physical characteristic.

4.10 PASSWORDS (Weak Authentication)

The simplest and oldest method of entity authentication is the password-based authentication, where the password is something that the claimant knows.

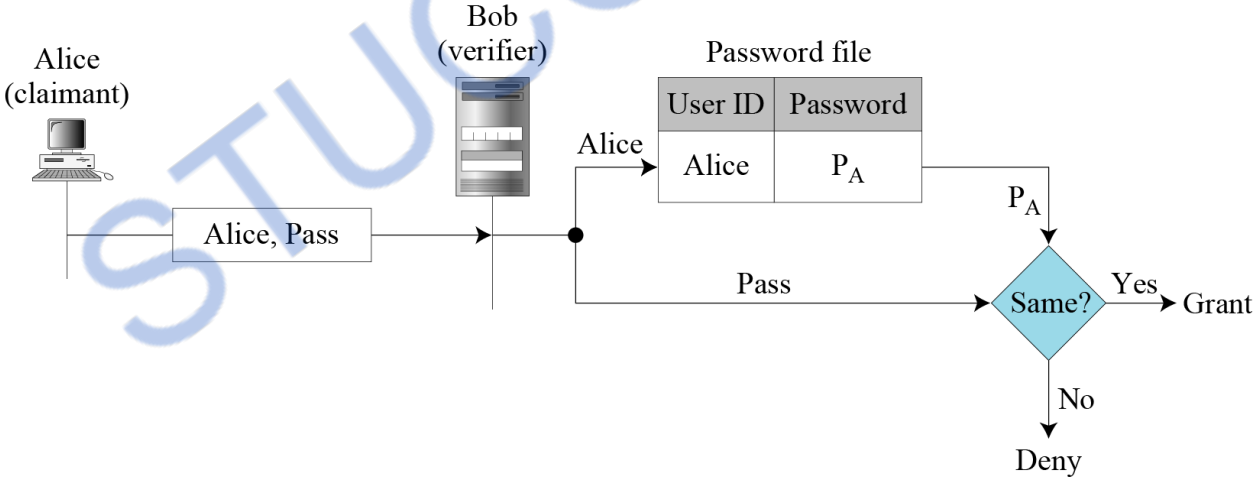
This is amongst the most conventional schemes where in a user has an „user id" and a „password". User id acts like a claim and password as evidence supporting the claim.
The system checks to see if it matches or not. Here demonstration of knowledge of the secret which is password in this case; corroborates that the person is verified.

Passphrase :

- The user types in a sentence rather than a short word for password. The idea is it is easier for a user to remember sentences.
- The entered sentence is then compared with what is stored in for the corresponding user.
- The long password can be encrypted or in plain-text when stored on the medium or data base

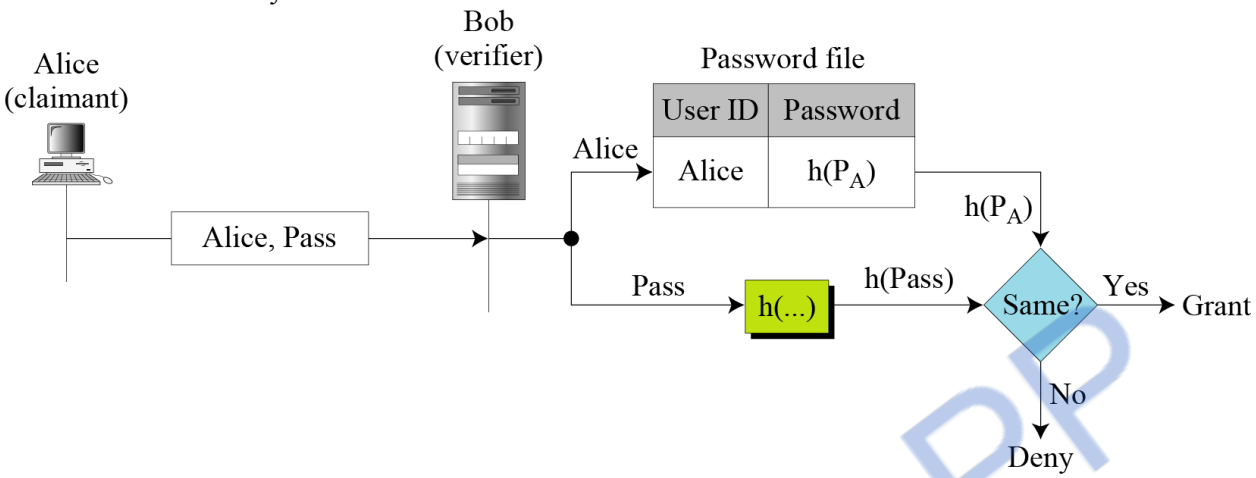
First Approach User ID and password file

P_A: Alice's stored password
Pass: Password sent by claimant



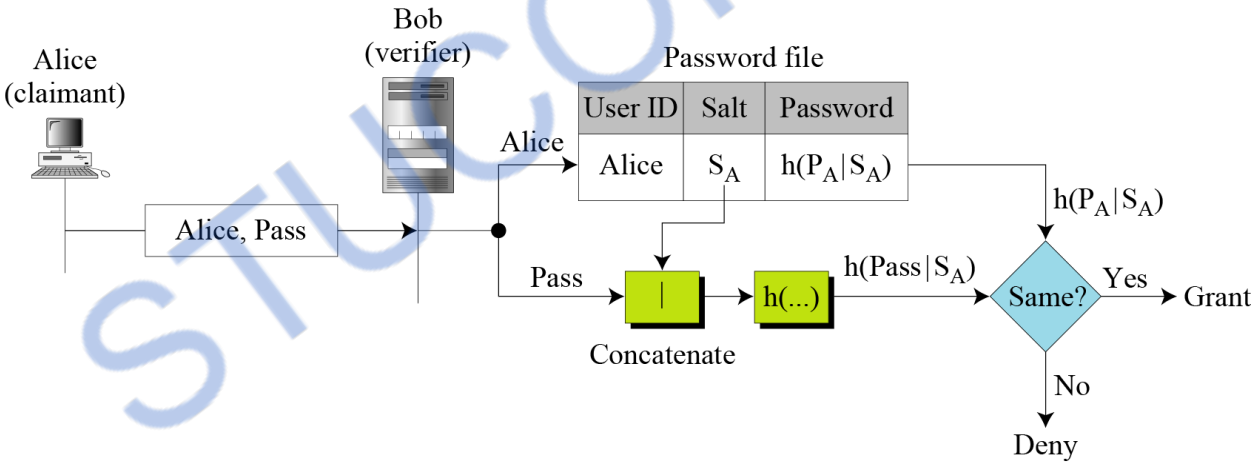
Second Approach Hashing the password

P_A: Alice's stored password
Pass: Password sent by claimant



Third Approach Salting the password

P_A: Alice's password
S_A: Alice's salt
Pass: Password sent by claimant



Fourth Approach

In the fourth approach, two identification techniques are combined. A good example of this type of authentication is the use of an ATM card with a PIN (personal identification number).

Advantages :

- It has better entropy than a short password.
- It is easier to remember than the usual passwords.

Disadvantage:

- This is really weak against attacks as intruder can hear over communication channel and impersonate it later.
- It is also very easy to replay the same message and use it later.
- Exhaustive search or password guessing i.e. dictionary attacks can also be used.
- One has to type extra.

4.11 CHALLENGE-RESPONSE (Strong Auhenticaion)

In password authentication, the claimant proves her identity by demonstrating that she knows a secret, the password.

In challenge-response authentication, the claimant proves that she knows a secret without sending it.

The central idea of challenge response is that claimant proves its identity to verifier by demonstrating knowledge of a secret known to be associated with entity without revealing the secret itself to the verifier during the protocol.

The challenge is usually time variant and is random number.

As every time the challenge is different, even if the adversary is monitoring the network it won‟t help as challenge changes every time.

Note

In challenge-response authentication, the claimant proves that she knows a secret without sending it to the verifier.

Note

The challenge is a time-varying value sent by the verifier; the response is the result of a function applied on the challenge.

Using a Symmetric-Key Cipher

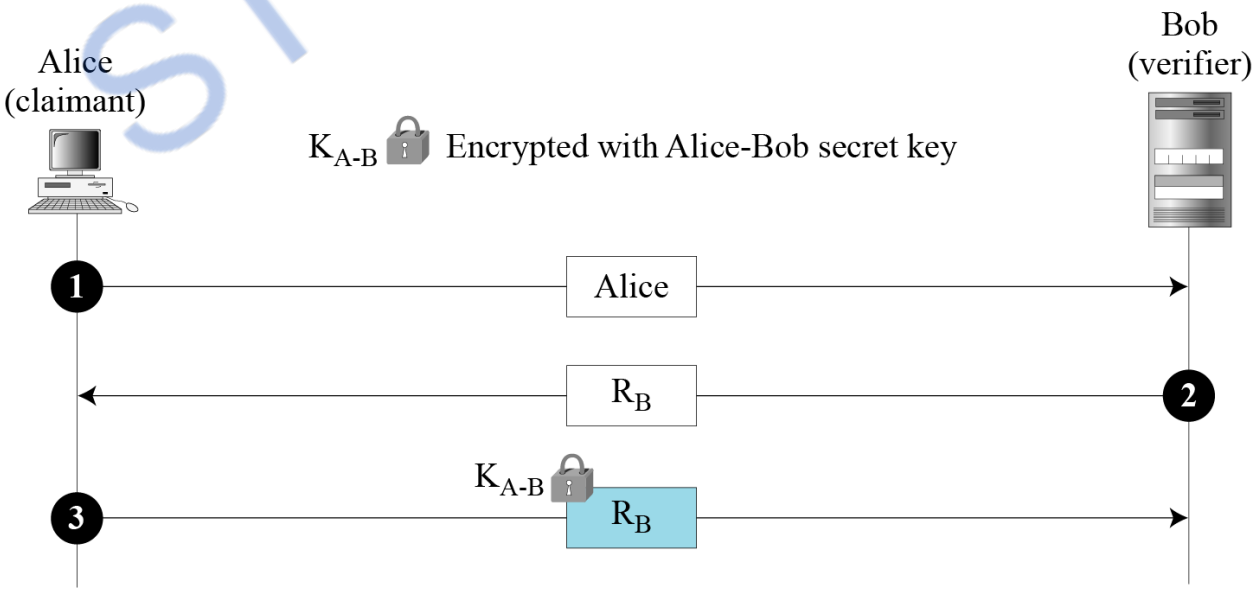
Challenge-response by symmetric-key techniques -unilateral authentication, using random number.

- Here, A is the claimant and B is the verifier.
- Communication takes place as

A →B : r_B----(1)
A →B : EK(r_B, B*) ----(2)
B sends A a random number. To prove its claim, A then encrypts the random number send by B using the symmetric encryption key „k". It also sends the optional field of the verifier as „B". This prevents reflection attack as the key used is bi-directional key „k".
B then decrypts the message sent by A to see the random number is the same as it had sent. It also sees if the identifier matches. If either of them is not true it stops any further communication.

First Approach Unidirectional Authentication

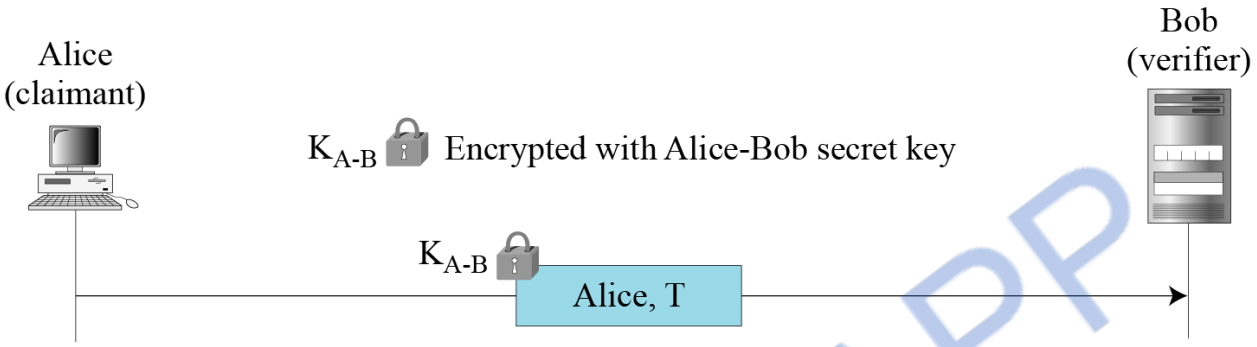
Nonce challenge



- Cannot be replaced by EVE

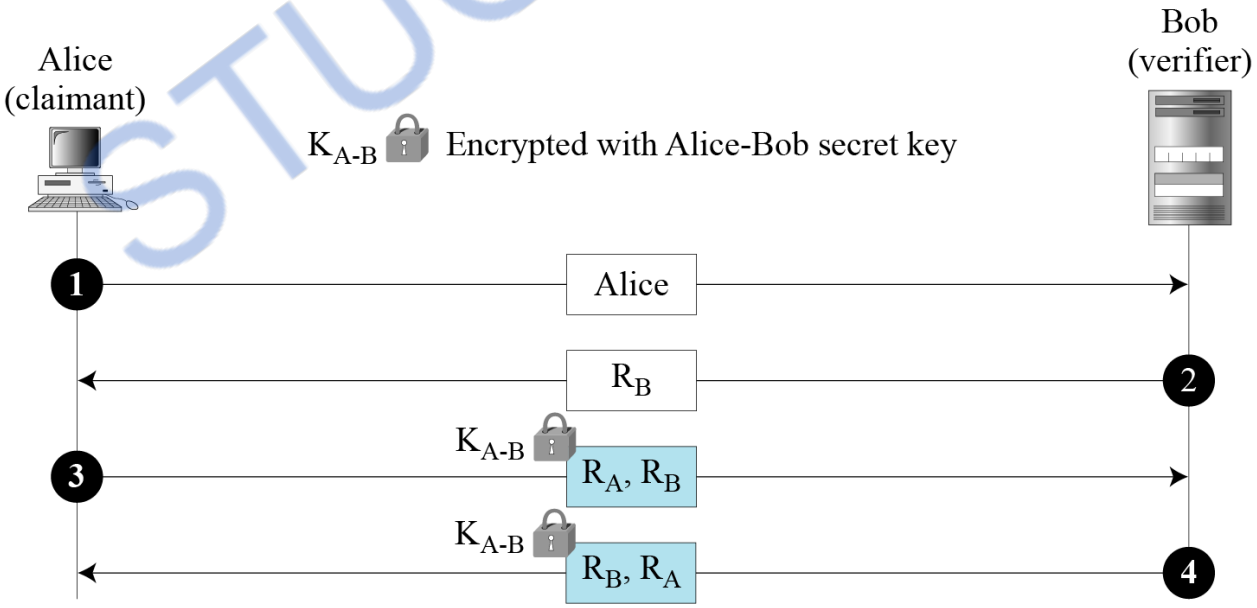
Second Approach Unidirectional Authentication

Timestamp challenge



- Assumed that the clocks are synchronized.
- Authentication can be done with only one message

Third Approach Bidirectional authentication

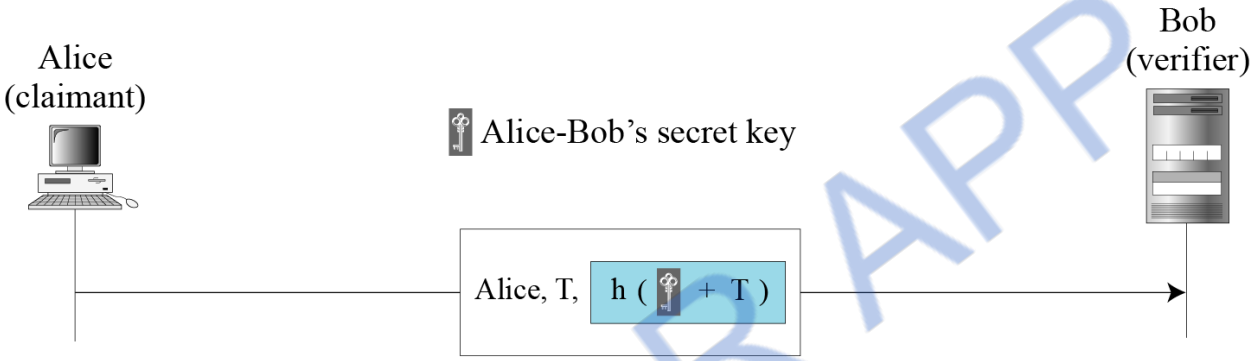


Order of RA and RB are changed to prevent Replay attack of the third message.

Using Keyed-Hash Functions

Instead of using encryption/decryption for entity authentication, we can also use a keyed-hash function (MAC).

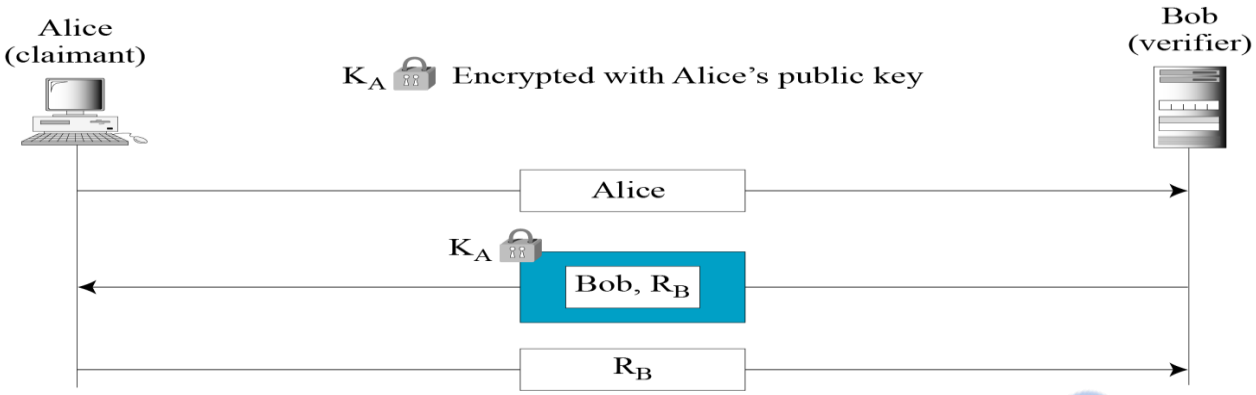
Keyed-hash function



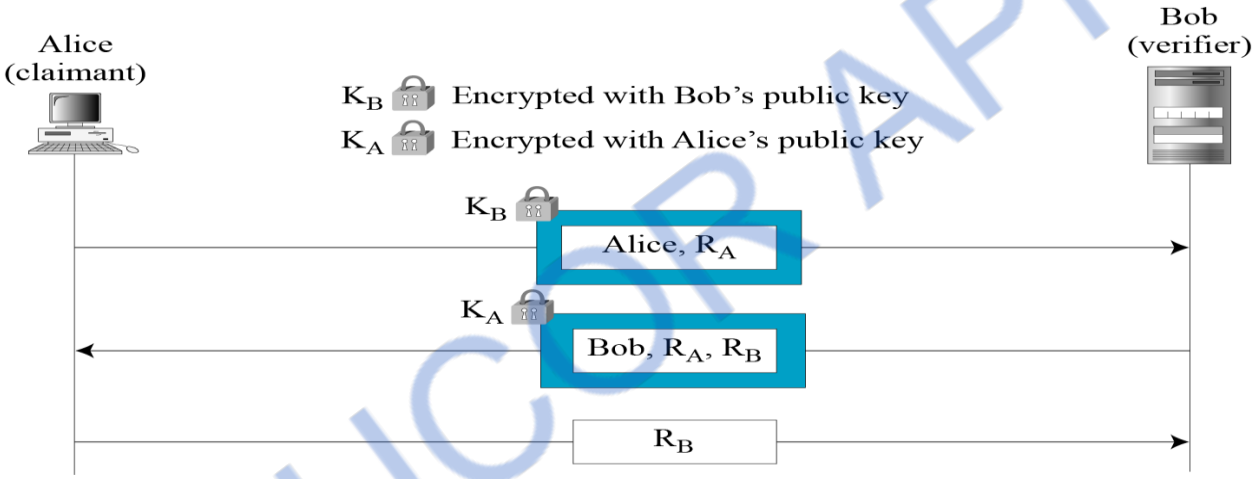
Using an Asymmetric-Key Cipher

- Verifier encrypts the challenge with the Public key of the claimant.
- Claimant decrypts the challenge with her private key.
- Sends the challenge back to the verifier.

First Approach Unidirectional, asymmetric-key authentication



Second Approach Bidirectional, asymmetric-key



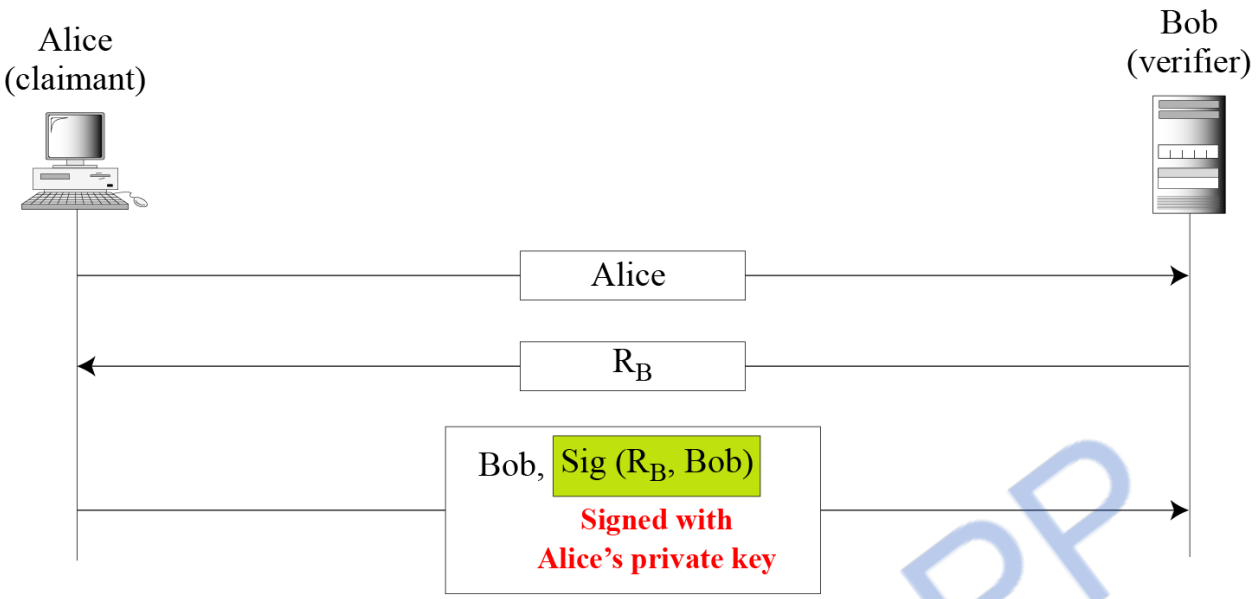
Bob is Authenticated

Alice is Authenicated

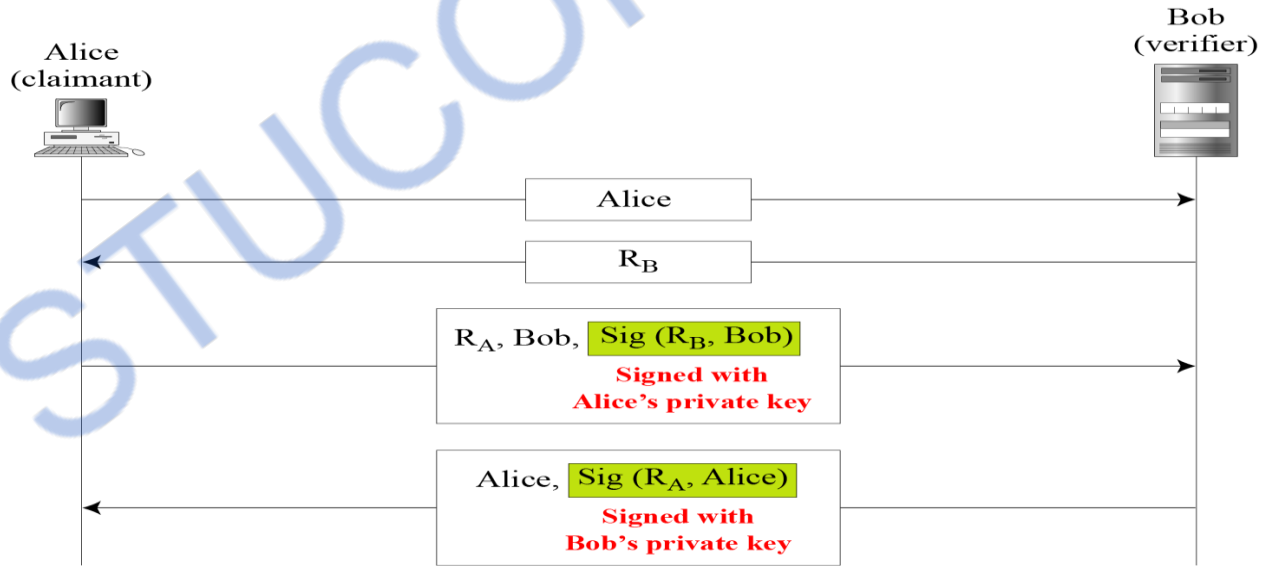
Using Digital Signature

Claimant uses private key for signing.

First Approach Digital signature, Unidirectional



Second Approach Digital signature, bidirectional authentication



Advantages::

- It is a time variant challenge where the response depends on entity‟s secret and challenge.
- Even if the communication line is monitored, the response from one execution of identification protocol will not provide an adversary with useful information.

Disadvantages::

- This protocol still would reveal some partial information about the claimant‟s secret, an adversarial verifier might still be able to strategically select challenges to obtain responses providing information.

4.12 BIOMETRICS

- Biometrics is the measurement of physiological or behavioral features that identify a person (authentication by something inherent).
- Biometrics measures features that cannot be guessed, stolen, or shared.

Components

Several components are needed for biometrics, including capturing devices, processors, and storage devices.

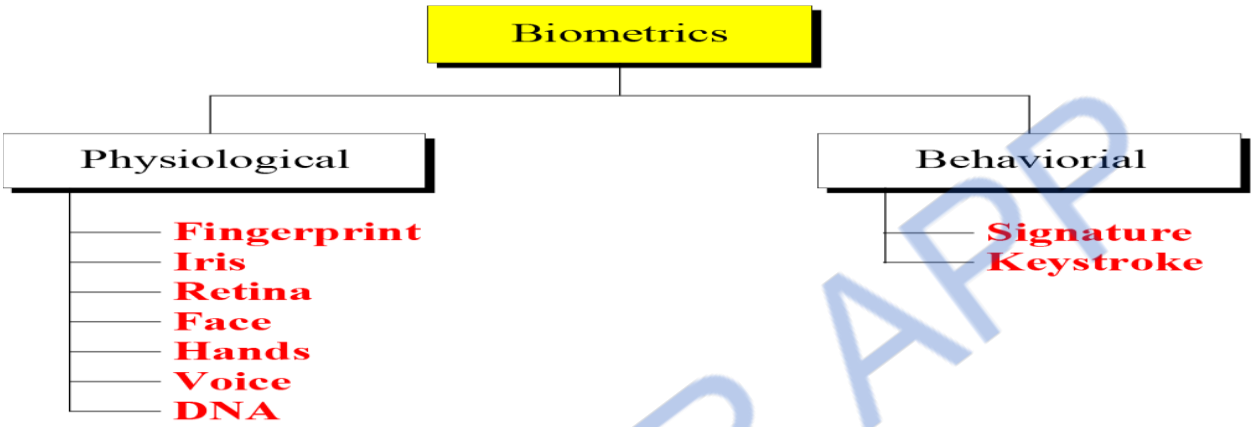
Enrollment

Before using any biometric techniques for authentication, the corresponding feature of each person in the community should be available in the database. This is referred to as enrollment.

Authentication

- Verification
- Identification

Techniques



Physiological Techniques

- Fingerprint
- Iris Retina
- Face Hands
- Voice DNA

Behavioral Techniques

- Signature
- Keystroke

Applications

Several applications of biometrics are already in use.

- In commercial environments, these include access to facilities, access to information systems, transaction at point-of-sales, and employee timekeeping.
- In the law enforcement system, they include investigations (using fingerprints or DNA) and forensic analysis.
- Border control and immigration control also use some biometric techniques. Applications

4.13Authentication Application

Kerberos

Kerberos is an authentication service.

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

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

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

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

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

Kerberos Version 4

Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service.

A Simple Authentication Dialogue

(1) C → AS: $ID_C \| P_C \| ID_V$

(2) A → C: Ticket

(3) C → V: $ID_C \|$ Ticket

Ticket = $E(K_v, [ID_C \| AD_C \| ID_V])$

where

C = client

AS = authentication server

V = server

ID_C = identifier of user on C

ID_V = identifier of V

P_C = password of user on C

AD_C = network address of C
K_v = secret encryption key shared by AS and V

A More Secure Authentication Dialogue

Introduce a scheme for avoiding plaintext passwords and a new server, known as the **ticket-granting server** (TGS).

Once per user logon session:
(1) C → AS: $ID_C \| IDt_{gs}$
(2) AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:
(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$
(4) TGS → C: $Ticket_v$

Once per service session:
(5) C → V: $ID_C \| Ticket_v$
$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime1])$
$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime2])$

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password (K_c), which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS (K_{tgs}) and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the servicegranting ticket. The server authenticates by using the contents of the ticket.

Authentication Service Exchange to obtain ticket-granting ticket

(1) $C \rightarrow AS \quad ID_c \| ID_{tgs} \| TS_1$

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

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

Ticket-Granting Service Exchange to obtain service-granting ticket

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

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

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

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

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

\

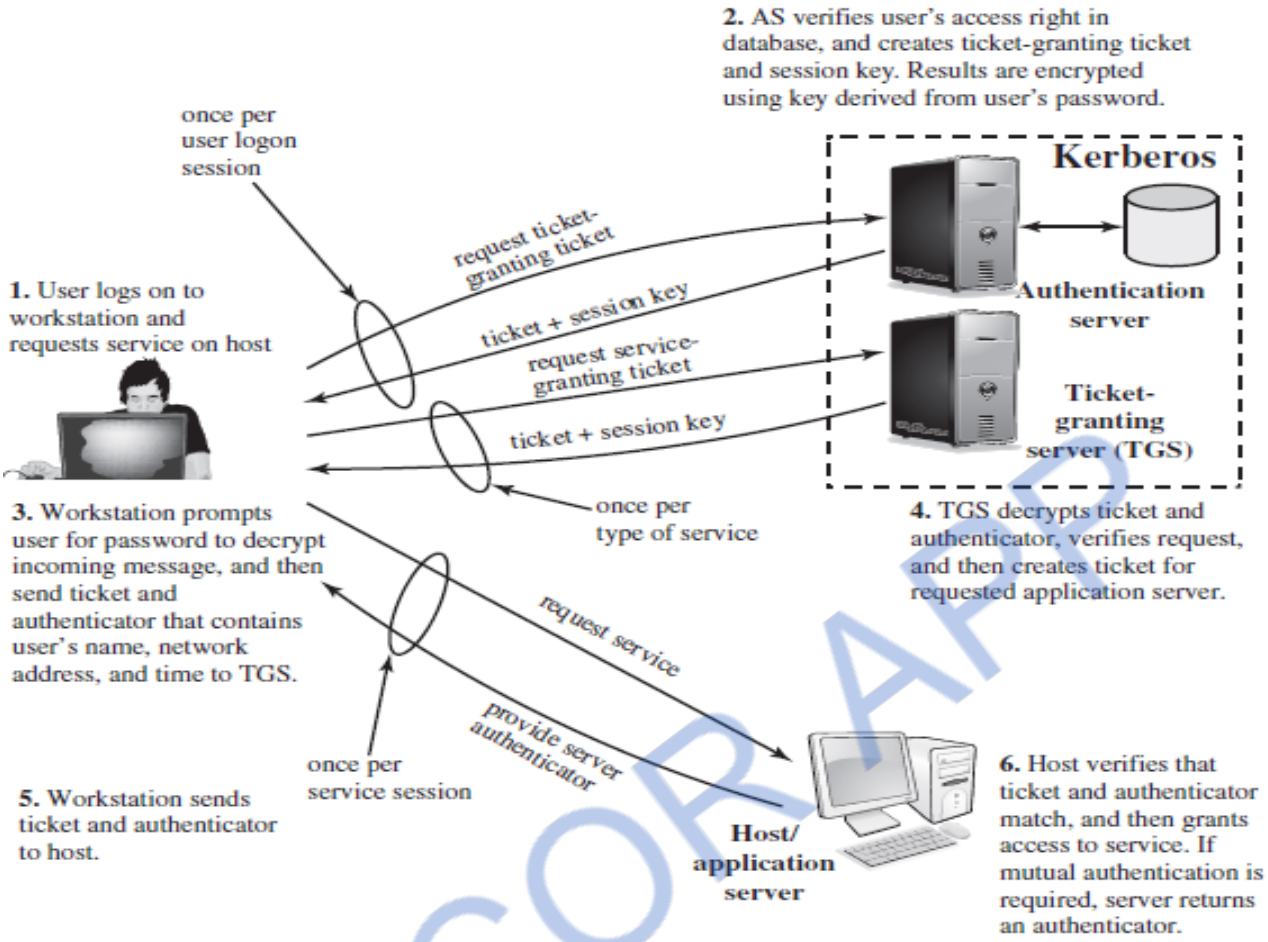
Client/Server Authentication Exchange to obtain service

(5) $C \rightarrow V \quad Ticket_v \| Authenticator_c$

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

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

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



Kerberos Realms and Multiple Kerberi

A full-service 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.

Such an environment is referred to as a **Kerberos realm**. The concept of **realm** can be explained as follows.

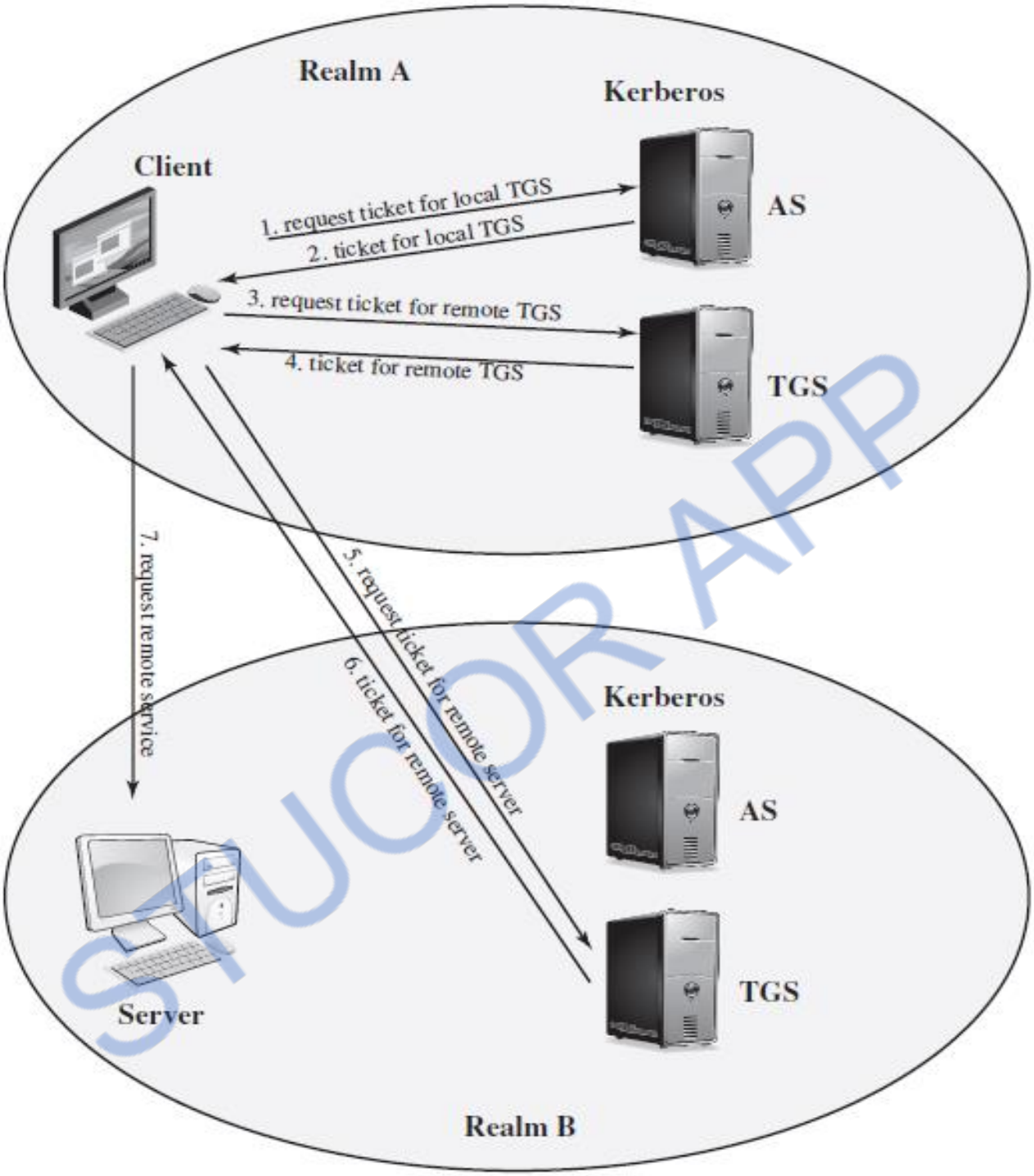
- A Kerberos realm is a set of managed nodes that share the same Kerberos database.
- The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room.
- A readonly copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system.

- Changing or accessing the contents of a Kerberos database requires the Kerberos master password. A related concept is that of a **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.

Kerberos provides a mechanism for supporting such interrealm authentication. For two realms to support interrealm authentication, a third requirement is added:

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.

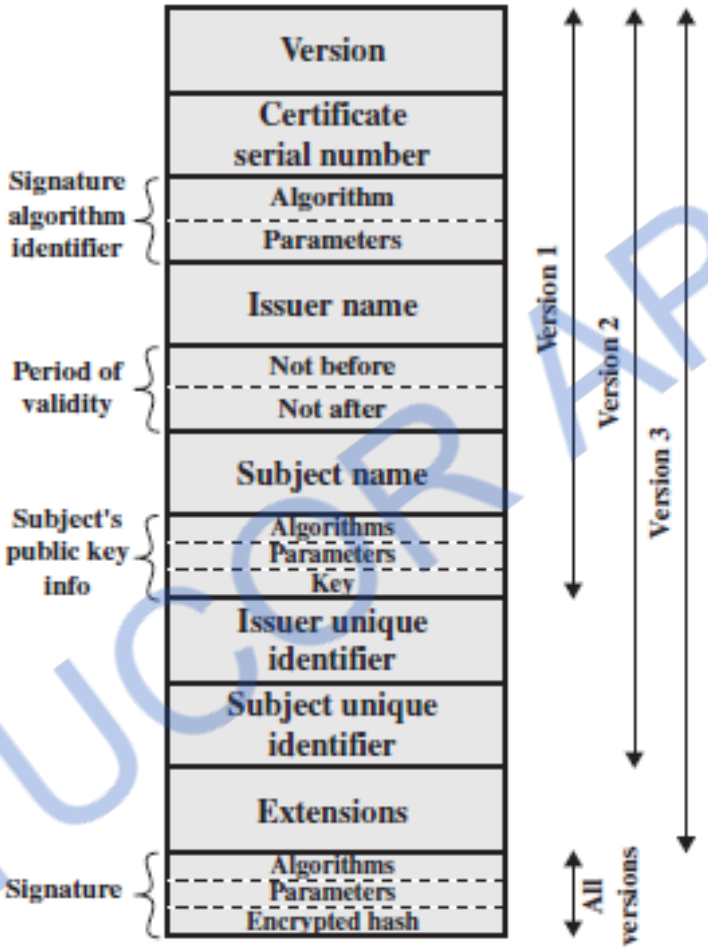
(1) $C \rightarrow AS: ID_c \ || \ ID_{tgs} \ || \ TS_1$
(2) $AS \rightarrow C: E(K_c, [K_{c,\ tgs} \ || \ ID_{tgs} \ || \ TS_2 \ || \ Lifetime_2 \ || \ Ticket_{tgs}])$
(3) $C \rightarrow TGS: ID_{tgsrem} \ || \ Ticket_{tgs} \ || \ Authenticator_c$
(4) $TGS \rightarrow C: E(K_{c,tgs,} [K_{c,\ tgsrem} \ || \ ID_{tgsrem} \ || \ TS_4 \ || \ Ticket_{tgsrem}])$
(5) $C \rightarrow TGS_{rem}: ID_{vrem} \ || \ Ticket_{tgsrem} \ || \ Authenticator_c$
(6) $TGS_{rem} \rightarrow C: E(K_{c,tgsrem,} [K_{c,\ vrem} \ || \ ID_{vrem} \ || \ TS_6 \ || \ Ticket_{vrem}])$
(7) $C \rightarrow V_{rem}: Ticket_{vrem} \ || \ Authenticator_c$



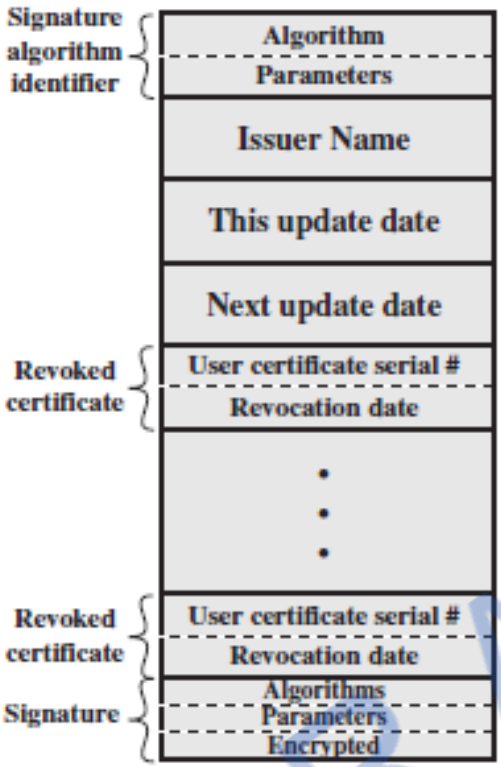
Kerberos Realms and Multiple Kerberi

4.14 X.509 certificate

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



(a) X.509 certificate



(b) Certificate revocation list

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

includes the signature algorithm identifier.

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

$$CA << A >> = CA \{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

where
Y <<X>> = the certificate of user X issued by certification authority Y
Y {I} = the signing of I by Y. It consists of I with an encrypted hash code appended
V = version of the certificate
SN = serial number of the certificate
AI = identifier of the algorithm used to sign the certificate
CA = name of certificate authority
UCA = optional unique identifier of the CA
A = name of user A
UA = optional unique identifier of the user A
Ap = public key of user A
T^A = period of validity of the certificate

Obtaining a User's Certificate
User certificates generated by a CA have the following characteristics:
• Any user with access to the public key of the CA can verify the user public key that was certified.
• No party other than the certification authority can modify the certificate without this being detected.

A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2. If A does not securely know the public key of X_2, then B's certificate, issued by X_2, is useless to A. A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.

Step 1 A obtains from the directory the certificate of X_2 signed by X_1. Because A securely knows X_1's public key, A can obtain X_2's public key from its certificate and verify it by means of X_1's signature on the certificate.
Step 2 A then goes back to the directory and obtains the certificate of B signed by X_2. Because A now has a trusted copy of X_2's public key, A can verify the signature and securely obtain B's public key.

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

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

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

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

This scheme need not be limited to a chain of two certificates. An arbitrarily

long path of CAs can be followed to produce a chain. A chain with *N* elements would be expressed as

$$X_1 << X_2 >> X_2 << X_3 >> .... X_N << B >>$$

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

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

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

X.509 Version 3

The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed.

1. The subject field is inadequate to convey the identity of a key owner to a public- key user. X.509 names may be relatively short and lacking in obvious identification details that may be needed by the user.
2. The subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other Internetrelated identification.
3. There is a need to indicate security policy information. This enables a security application or function, such as IPSec, to relate an X.509 certificate to a given policy.
4. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
5. It is important to be able to identify different keys used by the same owner at different times. This feature supports key lifecycle management: in particular, the ability to update key pairs for users and CAs on a regular basis or under exceptional circumstances.

Key and Policy Information

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy. A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.

- **Authority key identifier:** Identifies the public key to be used to verify the signature on this certificate or CRL. Enables distinct keys of the same CA to be differentiated. One use of this field is to handle CA key pair updating.
- **Subject key identifier:** Identifies the public key being certified. Useful for subject key pair updating. Also, a subject may have multiple key pairs and, correspondingly, different certificates for different purposes (e.g., digital signature and encryption key agreement).
- **Key usage:** Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used. May indicate one or more of the following: digital

signature, nonrepudiation, key encryption, data encryption, key agreement, CA signature verification on certificates, CA signature verification on CRLs.

• **Private-key usage period:** Indicates the period of use of the private key corresponding to the public key. Typically, the private key is used over a different period from the validity of the public key. For example, with digital signature keys, the usage 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. This extension lists policies that the certificate is recognized as supporting, together with optional qualifier information.

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

Certificate Subject and Issuer Attributes

These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject to increase a certificate user's confidence that the certificate subject is a particular person or entity.

The extension fields in this area include:

• **Subject alternative name:** Contains one or more alternative names, using any of a variety of forms. This field is important for supporting certain applications, such as electronic mail, EDI, and IPSec, which may employ their own name forms.

• **Issuer alternative name:** Contains one or more alternative names, using any of a variety of forms.

• **Subject directory attributes:** Conveys any desired X.500 directory attribute values for the subject of this certificate.

Certification Path Constraints

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

The extension fields in this area include:

• **Basic constraints:** Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.

• **Name constraints:** Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.

• **Policy constraints:** Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 5 NOTES

UNIT V SECURITY PRACTICE AND SYSTEM SECURITY

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

5.1 ELECTRONIC MAIL SECURITY

PRETTY GOOD PRIVACY (PGP)

PGP provides the confidentiality and authentication service that can be used for electronic mail and file storage applications.

The steps involved in PGP are

- Select the best available cryptographic algorithms as building blocks.
- Integrate these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy-touse commands.
- Make the package and its documentation, including the source code, freely available via the internet, bulletin boards and commercial networks.
- Enter into an agreement with a company to provide a fully compatible, low cost commercial version of PGP.

PGP has grown explosively and is now widely used.

A number of reasons can be cited for this growth.

- It is available free worldwide in versions that run on a variety of platform.
- It is based on algorithms that have survived extensive public review and are considered extremely secure.

e.g., RSA, DSS and Diffie Hellman for public key encryption

- It has a wide range of applicability.
- It was not developed by, nor it is controlled by, any governmental or standards organization.

Operational description

The actual operation of PGP consists of five services:

1. Authentication
2. Confidentiality
3. Compression
4. E-mail compatibility
5. Segmentation.

1. Authentication

The sequence for authentication is as follows:

- The sender creates the message
- SHA-1 is used to generate a 160-bit hash code of the message
- The hash code is encrypted with RSA using the sender‟s private key and the result is pretended to the message

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

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

2. Confidentiality

Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the conventional encryption algorithm CAST-128 may be used. The 64-bit cipher feedback (CFB) mode is used. In PGP, each conventional key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus although this is referred to as a session key, it is in reality a onetime key. To protect the key, it is encrypted with the receiver‟s public key.

The sequence for confidentiality is as follows:

- The sender generates a message and a random 128-bit number to be used as a session key for this message only.
- The message is encrypted using CAST-128 with the session key.
- The session key is encrypted with RSA, using the receiver‟s public key and is prepended to the message.
- The receiver uses RSA with its private key to decrypt and recover the session key.
- The session key is used to decrypt the message.

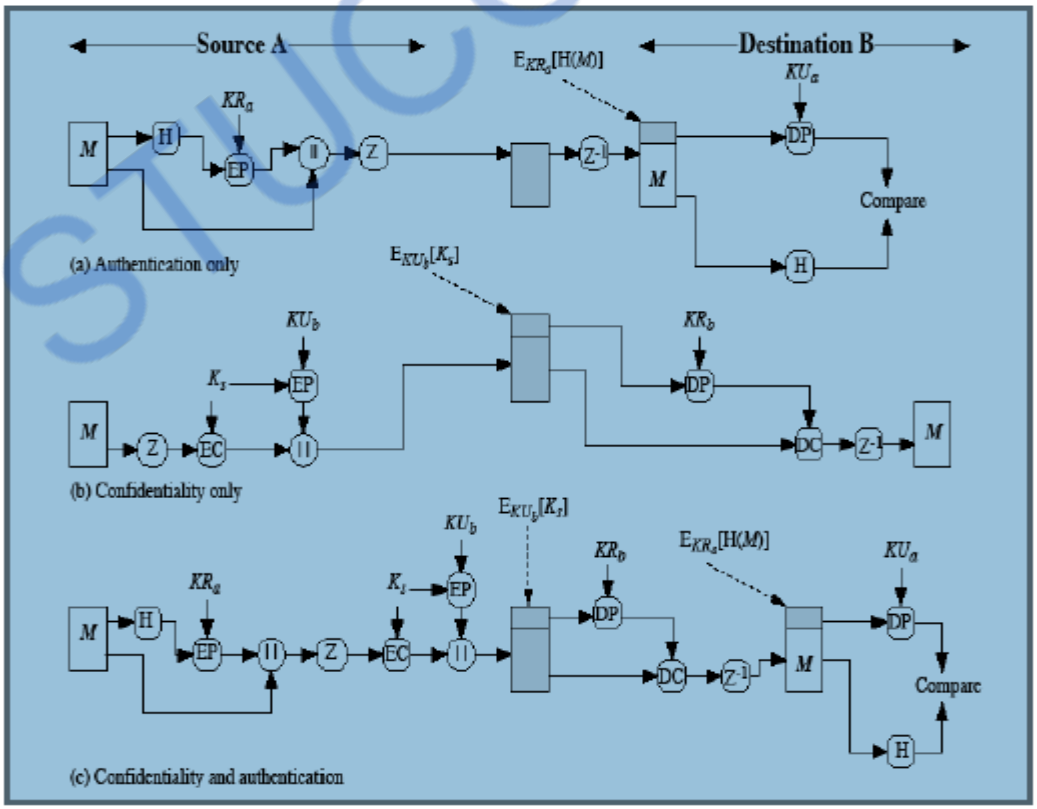
Confidentiality and authentication

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

3. Compression

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

- It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.
- Even if one were willing to generate dynamically a recompressed message from verification, PGP‟s compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and as a result, produce different compression forms.
- Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult. The compression algorithm used is ZIP



PGP Cryptographic functions

4. E-mail compatibility

Many electronic mail systems only permit the use of blocks consisting of ASCII texts. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data is mapped into four ASCII characters.

5. Segmentation and reassembly

E-mail facilities often are restricted to a maximum length. E.g., many of the facilities accessible through the internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately. To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation is done after all the other processing, including the radix-64 conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the other steps.

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.

a. Session key generation

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

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

b. Key identifiers

If multiple public/private key pair are used, then how does the recipient know which of the public keys was used to encrypt the session key? One simple solution would be to transmit the public key with the message but, it is unnecessary wasteful of space. Another solution would be to associate an identifier with each public key that is unique at least within each user. The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID. The key ID associated with each public key consists of its least significant 64 bits. i.e., the key ID of public key KUa is (KUa mod 264).

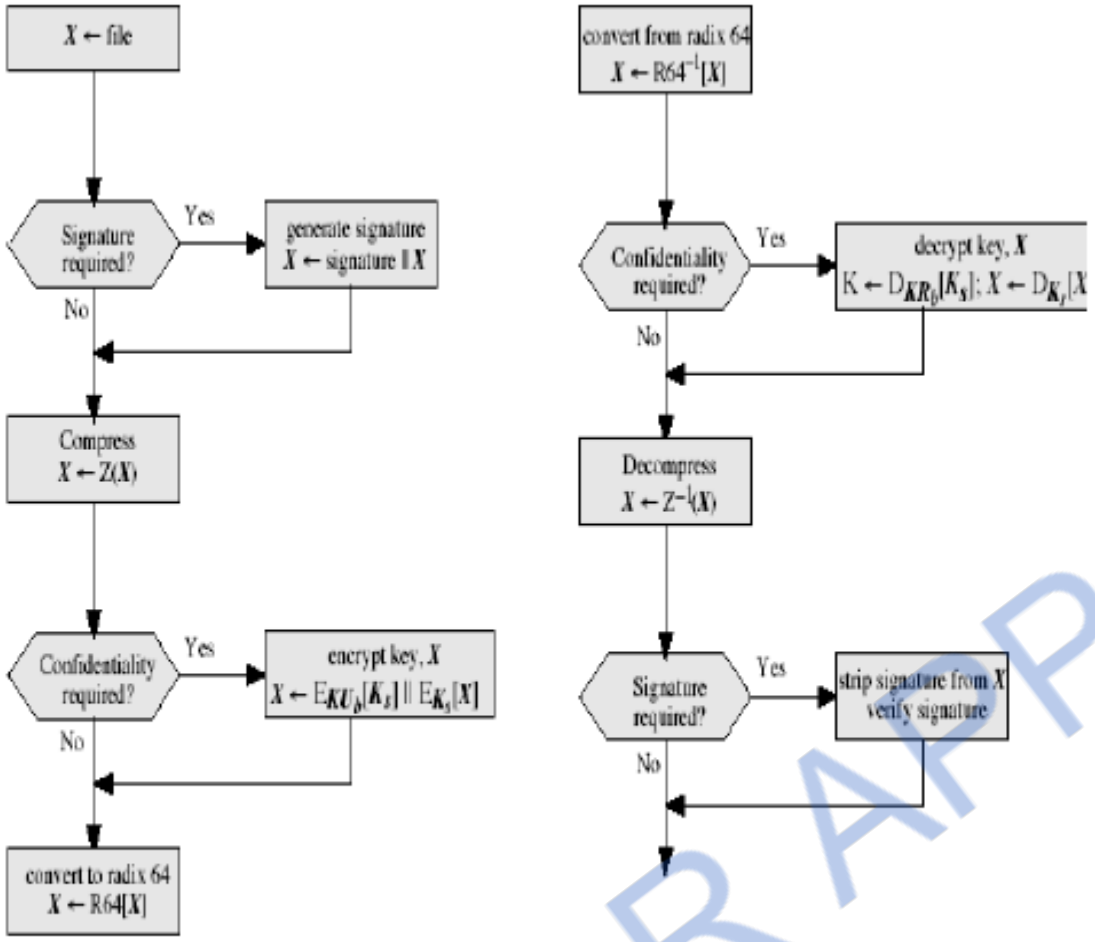
A message consists of three components.

- **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
- T**imestamp** – 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:

- EkUb= encryption with user B‟s Public key
- EKRa= encryption with user A‟s private key
- EKs = encryption with session key
- ZIP = Zip compression function
- R64 = Radix- 64 conversion function

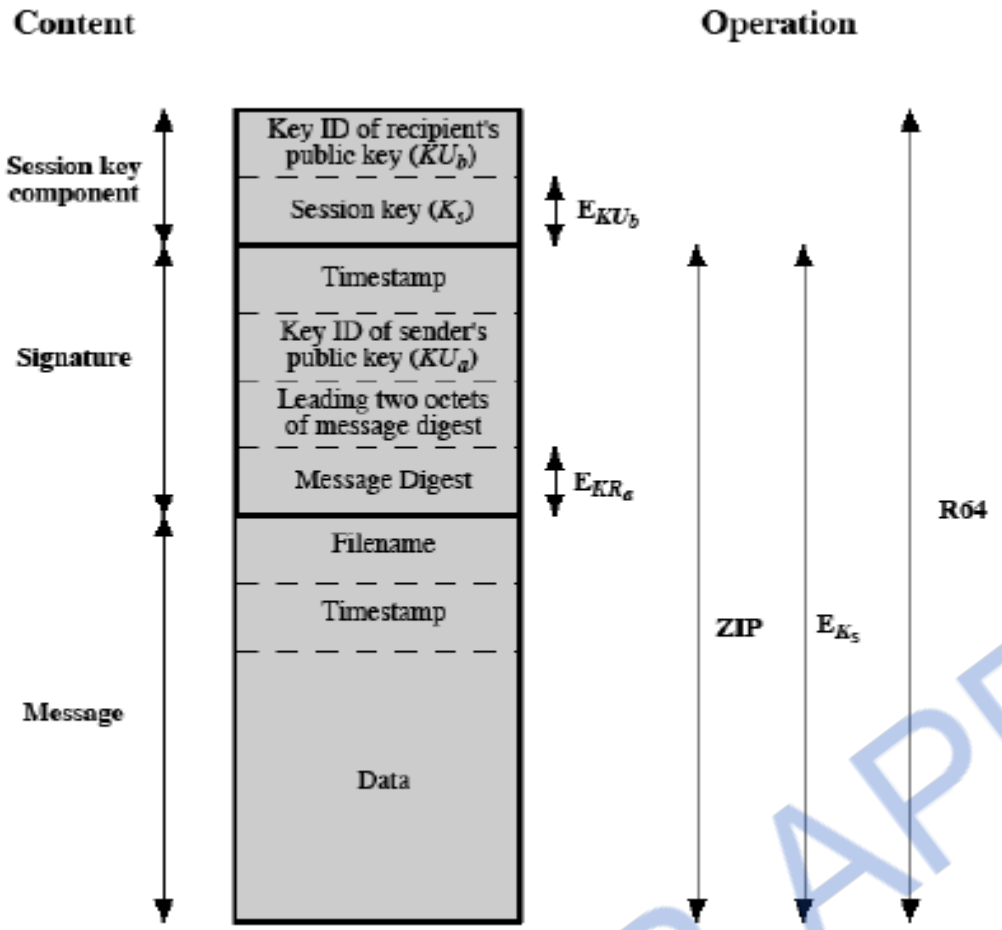


(a) Generic Transmission Diagram (from A) (b) Generic Reception Diagram (to B)

Transmission and Reception of PGP message

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.

Private Key Ring

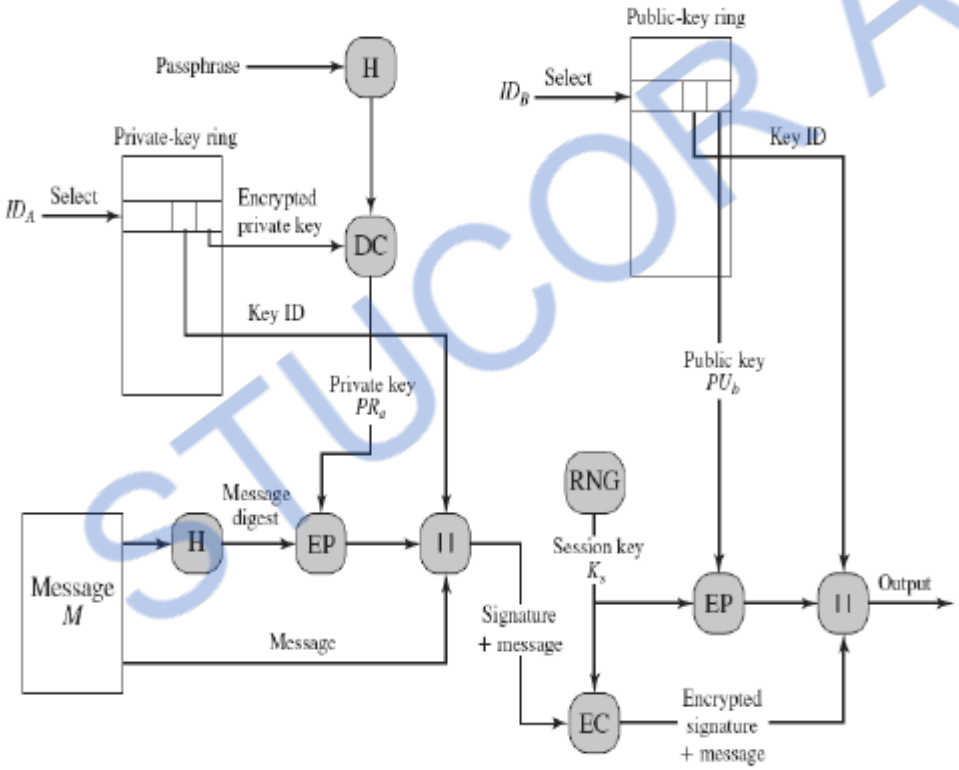
Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
. . .	. . .	. . .	. . .	. . .
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_s), PR_s)$	User i
. . .	. . .	. . .	. . .	. . .

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

* = field used to index table

General Structure public and private key



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.

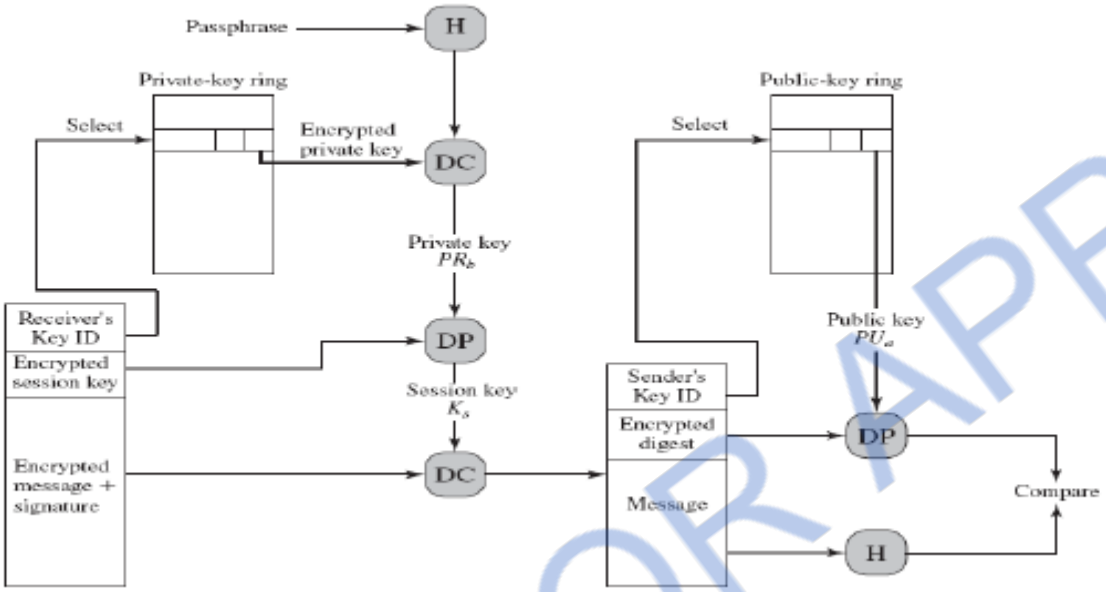
• PGP prompts the user for the passphrase (password) to recover the unencrypted private key.

• 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

5.2 S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security.

5.1.2.1 Multipurpose Internet Mail Extensions

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

Following are the limitations of SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects.
2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle non textual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

• Deletion, addition, or reordering of carriage return and linefeed

• Truncating or wrapping lines longer than 76 characters

• Removal of trailing white space (tab and space characters)

- Padding of lines in a message to the same length
- Conversion of tab characters into multiple space characters

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

OVERVIEW

The MIME specification includes the following elements:

1. **Five new message header** fields are defined, which may be included in an RFC 822 header. These fields provide information about the body of the message.
2. **A number of content formats** are defined, thus standardizing representations that support multimedia electronic mail.
3. **Transfer encodings** are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

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

The five header fields defined in MIME are as follows:

- **MIME-Version**: Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type**: Describes the data contained in the body with sufficient detail.
- **Content-Transfer-Encoding**: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description**: A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Content Types

There are seven different major types of content and a total of 15 subtypes

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.

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), defined in RFC 821, or some other mail transfer protocol and RFC 5322 for electronic mail. [PARZ06] lists the following limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/ UUdecode scheme. However, none of these is a standard or even a de facto standard.
2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

• Deletion, addition, or reordering of carriage return and linefeed
• Truncating or wrapping lines longer than 76 characters
• Removal of trailing white space (tab and space characters)
• Padding of lines in a message to the same length
• Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049.

Functions

• **Enveloped data:** This consists of encrypted content of any type and encryptedcontent encryption keys for one or more recipients.
• **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
• **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
• **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

Cryptographic Algorithms

• **MUST:** The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.

• **SHOULD:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Enhanced Security Services

• **Signed receipts:** A signed receipt may be requested in a SignedData object. Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message. In essence, the recipient signs the entire original message plus the original (sender's) signature and appends the new signature to form a new S/MIME message.

• **Security labels:** A security label may be included in the authenticated attributes of a SignedData object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitted access to an object. Other uses include priority (secret, confidential, restricted, and so on) or role based, describing which kind of people can see the information (e.g., patient's health-care team, medical billing agents, etc.).

• **Secure mailing lists:** When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA with encryption performed using the MLA's public key.

5.3 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 neighbour 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.

IP SECURITY ARCHITECTURE

Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology.

Encapsulating Security Payload (ESP):

Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.

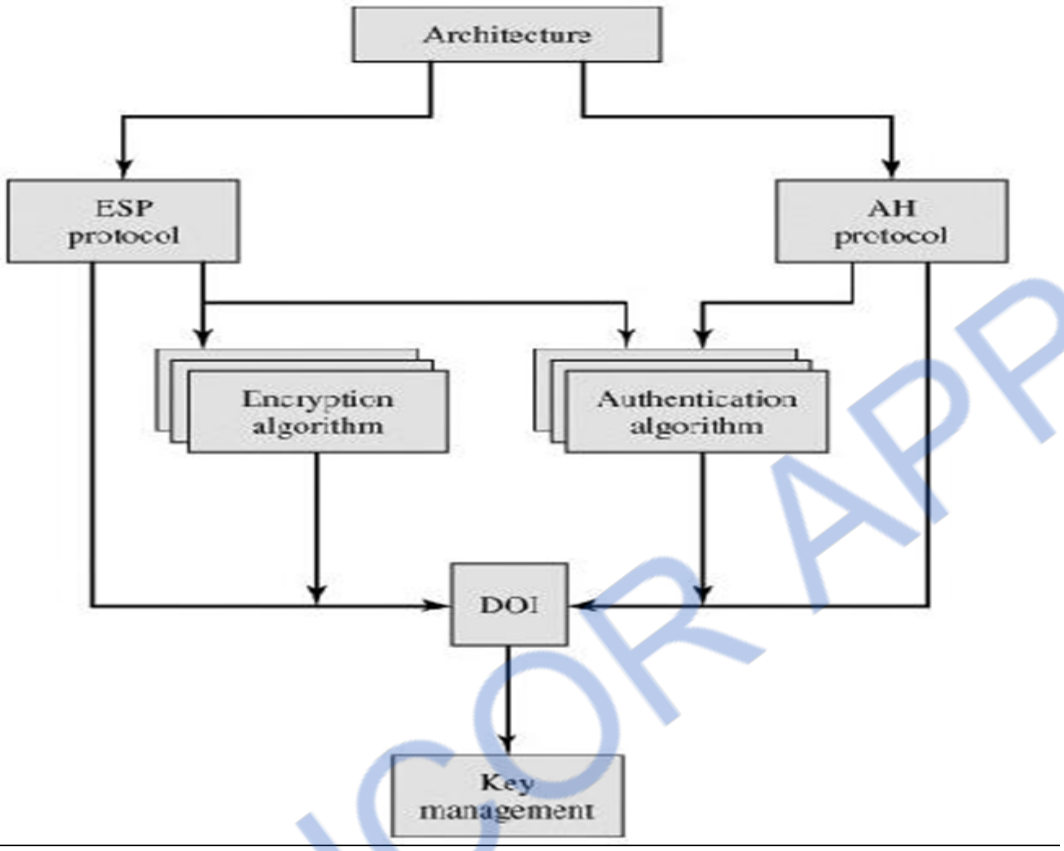
Authentication Header (AH):

Covers the packet format and general issues related to the use of AH for packet authentication.

Encryption Algorithm:

A set of documents that describe how various encryption algorithms are used for ESP.

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



IP security Document overview

IPSec Services

IPSec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services.

Two protocols are used to provide security:
- An authentication protocol: Designated by the header of the protocol, Authentication Header (AH);
- Encryption/authentication protocol designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP).

The services are

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

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:

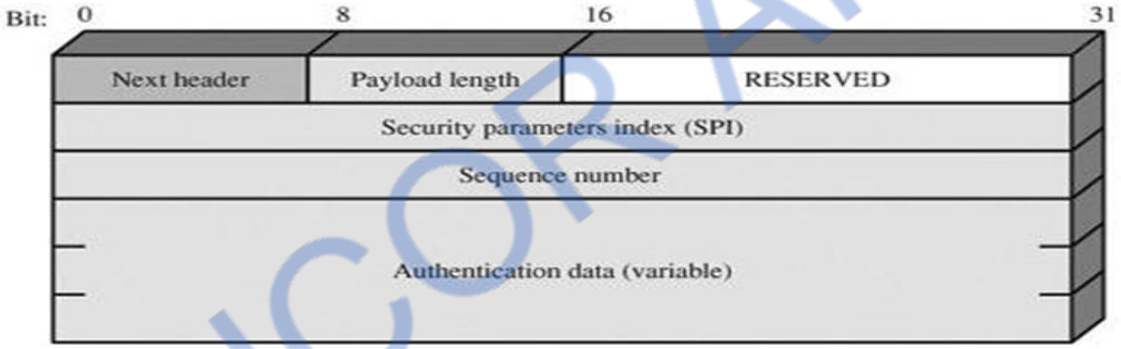
Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header.

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.

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.
- **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



KEY MANAGEMENT

The key management portion of IPSec involves the determination and distribution of secret keys. Two types of key management:

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

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

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

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

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

5.4 WEB SECURITY

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

Two types of attacks are:

Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted.

Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site.

SECURE SOCKET LAYER AND TRANSPORT LAYER SECURITY

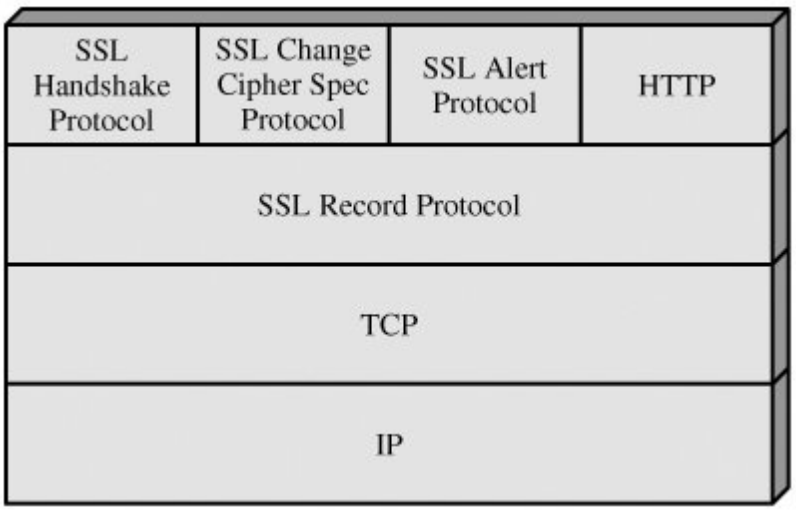
SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

Connection:

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



SSL PROTOCOL STACK

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 Record Protocol

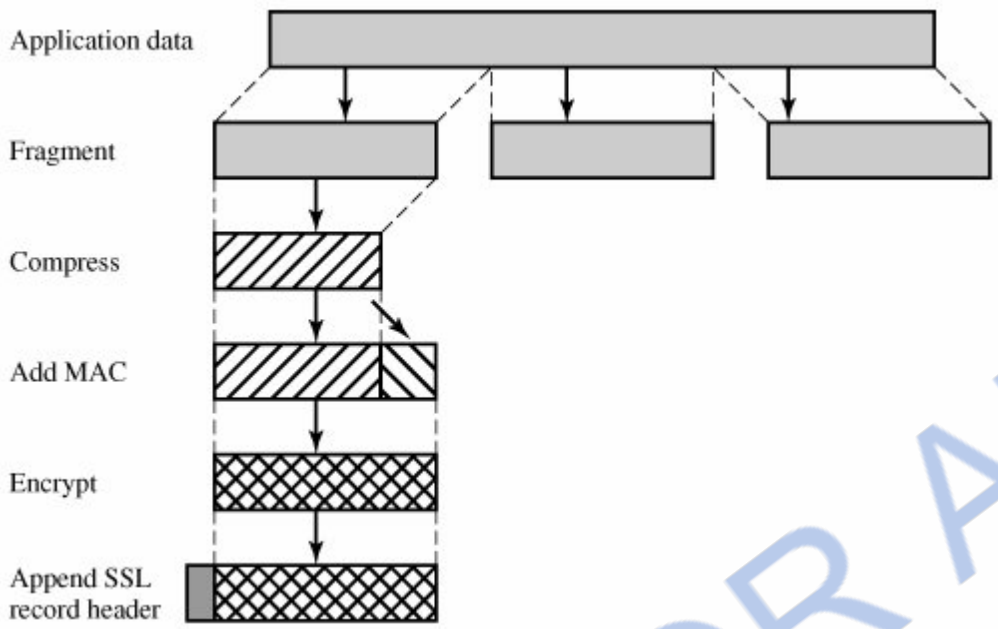
The SSL Record Protocol provides two services for SSL connections:
Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication 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 compressionalgorithm 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 ofthe 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 (orcompressed fragment if compression is used). The maximum value is 2_{14} + 2048.



SSL Record Protocol Operation

Change Cipher Spec Protocol

This protocol consists of a single message which consists of a single byte with the value

1. **Alert Protocol**

The Alert Protocol is used to convey SSL-related alerts to the peer entity.

2. **Handshake Protocol**

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

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.

SET provides three services:

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

Key Features of SET

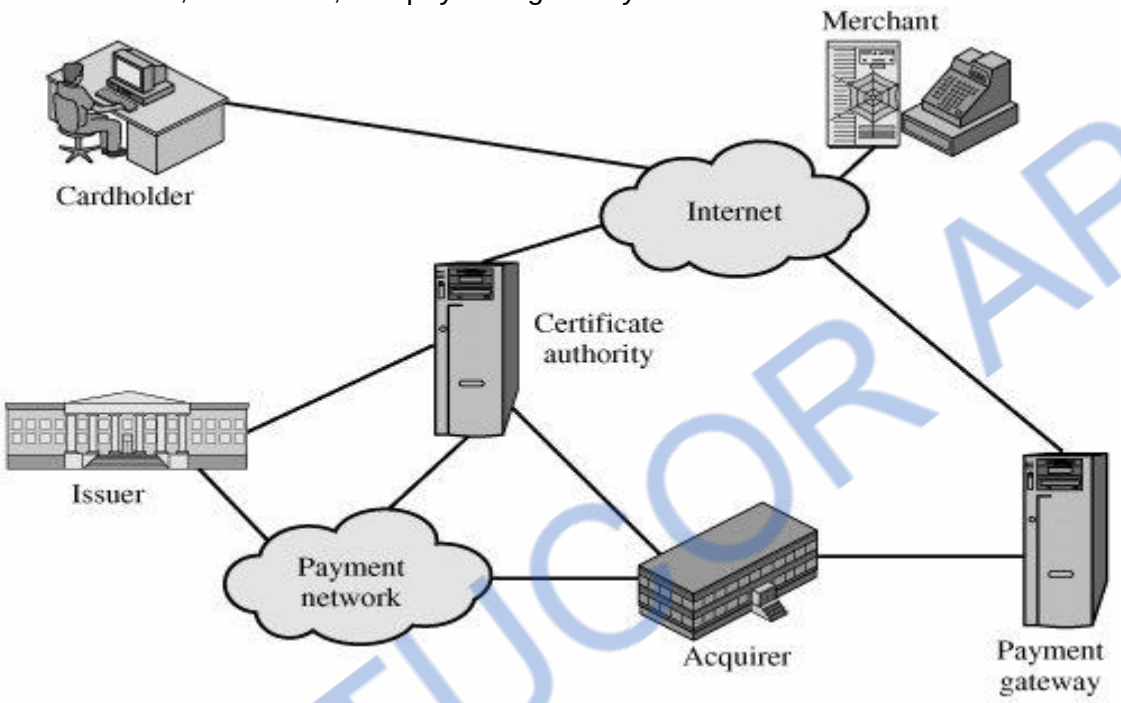
- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

SET Participants

- **Cardholder:** A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.
- **Merchant:** A merchant is a person or organization that has goods or services to sell to the cardholder.
- **Issuer**: This is a financial institution, such as a bank, that provides the cardholder with the payment card.
- **Acquirer**: This is a financial institution that establishes an ccount with a merchant andprocesses payment card authorizations and payments.

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

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



Secure Electronic Commerce Components

5.5 SYSTEM SECURITY

INTRUDERS

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

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

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.

Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However there is no way in advance to know whether an intruder will be benign or malign.

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

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

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

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

Intrusion Techniques:

The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruders to acquire information that should be protected. In most cases, the information is in the form of a user password. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. The password files can be protected in one of the two ways:

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

The following techniques are used for learning passwords.

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

Try words in the system"s online dictionary or a list of likely passwords. Collect informationabout users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.

- Try user"s phone number, social security numbers and room numbers.
- Try all legitimate license plate numbers.
- Use a Trojan horse to bypass restriction on access.
- Tap the line between a remote user and the host system.

Two principle countermeasures:

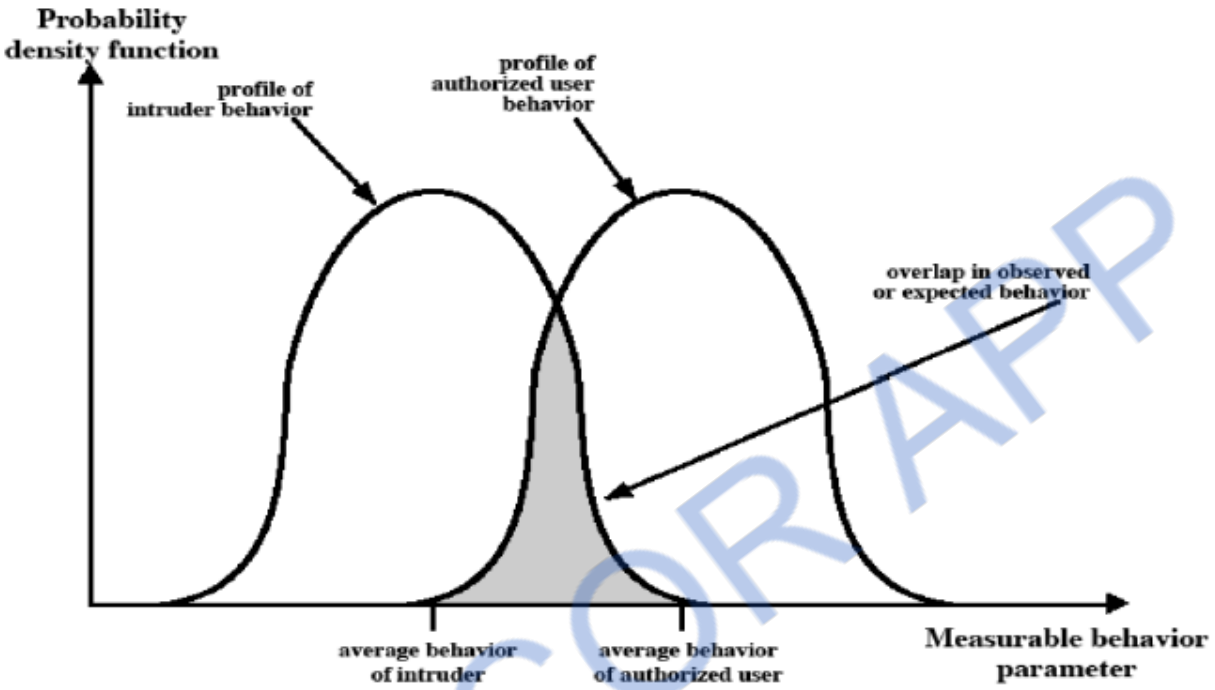
- **Detection** – concerned with learning of an attack, either before or after its success.
- **Prevention** – challenging security goal and an uphill bottle at all times.

Intrusion Detection

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years.
This interest is motivated by a number of considerations, including the following:
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 behaviour of the intruder differs from that of a legitimate user in ways that can be quantified. Figure suggests, in very abstract terms, the nature of the task onfront the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders.

On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of Compromise and art in the practice of intrusion detection



Profiles of Behavior of Intruders and Authorized Users

Approaches to Intrusion Detection

The approaches to Intrusion detection are,

- Statistical anomaly detection
- Rule-based detection

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

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.

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

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

Penetration identification: An expert system approach is used which searches for suspicious behavior.

Distributed Intrusion Detection

The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN points out the following major issues in the design of a distributed intrusion detection system.

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

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

The below diagram shows the overall architecture, which consists of three main components:

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

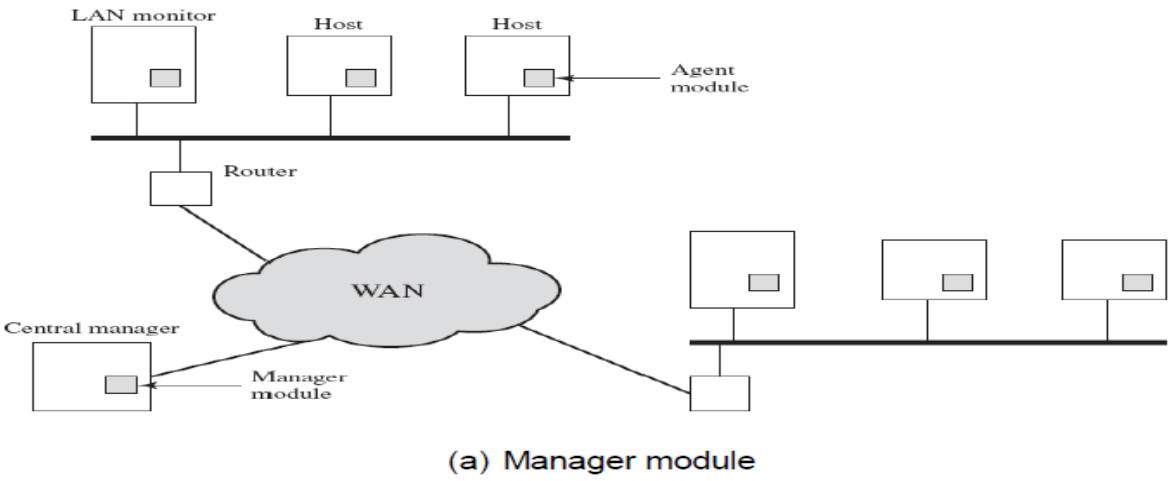
The scheme is designed to be independent of any operating system or system auditing Implementation

.

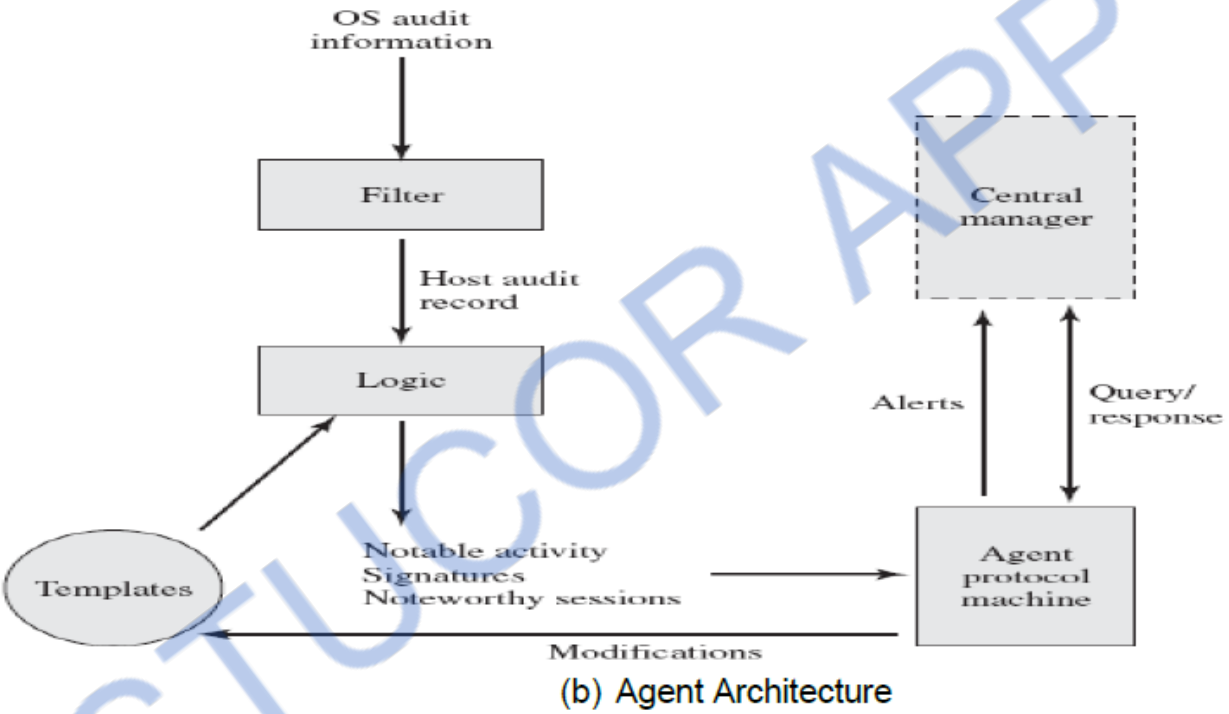
- The agent captures each audit record produced by the native audit collection system
- A filter is applied that retains only those records that are of security interest.
- These records are then reformatted into a standardized format referred to as the host audit record (HAR).

Next, a template-driven logic module analyzes the records for suspicious activity.

- At the lowest level, the agent scans for notable events that are of interest independent of any past events.
- Examples include failed file accesses, accessing system files, and changing a file's access control.
- At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures).
- Finally, the agent looks for anomalous historical profile of that user, such as number of programs executed, number of files accessed, and the like.
- When suspicious activity is detected, an alert is sent to the central manager.
- The central manager includes an expert system that can draw inferences from received data.
- The manager may also query individual systems for copies of HARs to correlate with those from other agents.
- The LAN monitor agent also supplies information to the central manager.
- The LAN monitor agent audits host-host connections, services used, and volume of traffic.
- It searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as rlogin.



(a) Manager module



(b) Agent Architecture

Architecture of DIDS

5.6 MALICIOUS SOFTWARE

The words “Malicious Software” coin the word “Malware” and the meaning remains the same. Malicious Software refers to any malicious program that causes harm to a computer system or network. Malicious Malware Software attacks a computer or network in the form of viruses, worms, trojans, spyware, adware or rootkits. Their mission is often targeted at accomplishing unlawful tasks such as robbing protected data, deleting confidential documents or add software without the user consent.

Different Types of Malicious Software

- Computer Virus

A computer virus is a malicious software which self-replicates and attaches itself to other files/programs. It is capable of executing secretly when the host program/file is activated. The different types of Computer virus are Memory-Resident Virus, Program File Virus, Boot Sector Virus, Stealth Virus, Macro Virus, and Email Virus.

- Worms

A worm is a malicious software which similar to that of a computer virus is a self-replicating program, however, in the case of worms, it automatically executes itself. Worms spread over a network and are capable of launching a cumbersome and destructive attack within a short period.

- Trojan Horses

Unlike a computer virus or a worm – the trojan horse is a non-replicating program that appears legitimate. After gaining the trust, it secretly performs malicious and illicit activities when executed. Hackers make use of trojan horses to steal a user's password information, destroy data or programs on the hard disk. It is hard to detect!

- Spyware/Adware

Spyware secretly records information about a user and forwards it to third parties. The information gathered may cover files accessed on the computer, a user's online activities or even user's keystrokes.
Adware as the name interprets displays advertising banners while a program is running. Adware can also work like spyware, it is deployed to gather confidential information. Basically, to spy on and gather information from a victim's computer.

- Rootkit

A rootkit is a malicious software that alters the regular functionality of an OS on a computer in a stealthy manner. The altering helps the hacker to take full control of the system and the hacker acts as the system administrator on the victim's system. Almost all the rootkits are designed to hide their existence.

Malicious Software History

Even before the internet became widespread, malicious software (virus) was infected on personal computers with the executable boot sectors of floppy disks. Initially, the computer viruses were written for the Apple II and Macintosh devices. After the IBM PC and MS-DOS system became more widespread they were also targeted in the similar fashion.
The first worms originated on multitasking Unix systems, they were the first network-borne infectious programs too. SunOS and VAX BSD systems were infected by the first well-known worm of the time called the Internet Worm of 1988. Ever since the advent of Microsoft Windows platform in the 1990s, the infectious codes were written in the macro language of Microsoft Word and similar programs.

Methods of protection against malicious software

Malicious Software is definitely a security threat for corporate users and individuals, thereby detecting and fighting malware remains on top of the agenda for many firms. Since the time BYOD culture started to flourish, Endpoint Security and Endpoint Protection have become the topics of discussion in many IT conference rooms. Many corporates today try to implement the best Endpoint Security or Endpoint Protection software to steer clear of the dangers.
Remember, if it is an individual system, it is essential to have an antivirus installed and if you already have one in place see to that it is updated at regular intervals. This approach will help you to remain safe during new breakouts. Comodo's Free Antivirus, Endpoint Security, Endpoint Protection Solutions are your best option for detecting and fighting malicious software. For more details visit our official page!

5.7 VIRUS AND RELATED THREATS

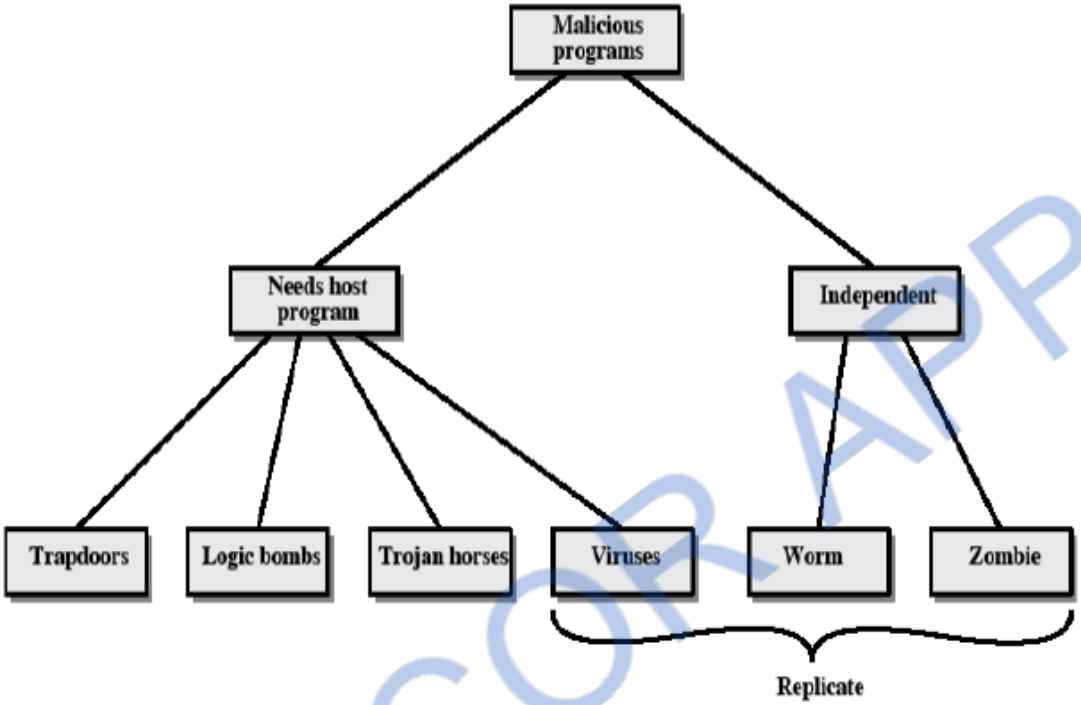
Malicious Programs:

Malicious software can be divided into **two categories**:

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

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples.

The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples



TAXONOMY OF MALICIOUS PROGRAMS

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.

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

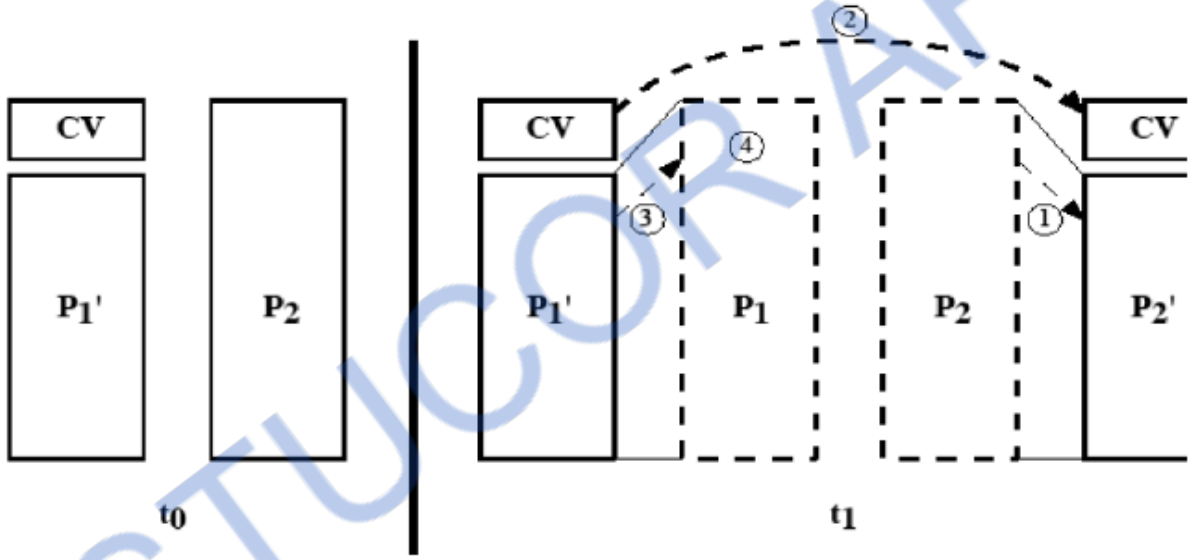
A virus can be pre pended or post pended to an executable program, or it can be embedded in some other fashion.

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.

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. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps

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



Compression Virus

Initial Infection:
Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes.
Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system.

Types of Viruses:
Following categories as being among the most significant types of viruses:
Parasitic virus: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.
Memory-resident virus: Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.

Boot sector virus: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
Stealth virus: A form of virus explicitly designed to hide itself from detection by antivirus software.
Polymorphic virus: A virus that mutates with every infection, making detection by the "signature" of the virus impossible.
Metamorphic virus: As with a polymorphic virus, a metamorphic virus 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.

Macro Viruses

Macro viruses are particularly threatening for a number of reasons:
1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.
2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file.

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
- The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package
- The virus does local damage

Worms

A worm is a 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. Network worm programs use network connections to spread from system to system.
Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions. 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:
- Search for other systems to infect by examining host tables or similar repositories of remote system addresses
- Establish a connection with a remote system
- 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

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.
1.It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems.

To obtain the passwords, the worm ran a passwordcracking program that tried

- Each user's account name and simple permutations of it
- A list of 432 built-in passwords that Morris thought to be likely candidates
- All the words in the local system directory

2.It exploited a bug in the finger protocol, which reports the location of a remote user.
3.It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

5.8 FIREWALLS

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 attacksand 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 asecure 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.

TYPES OF FIREWALLS

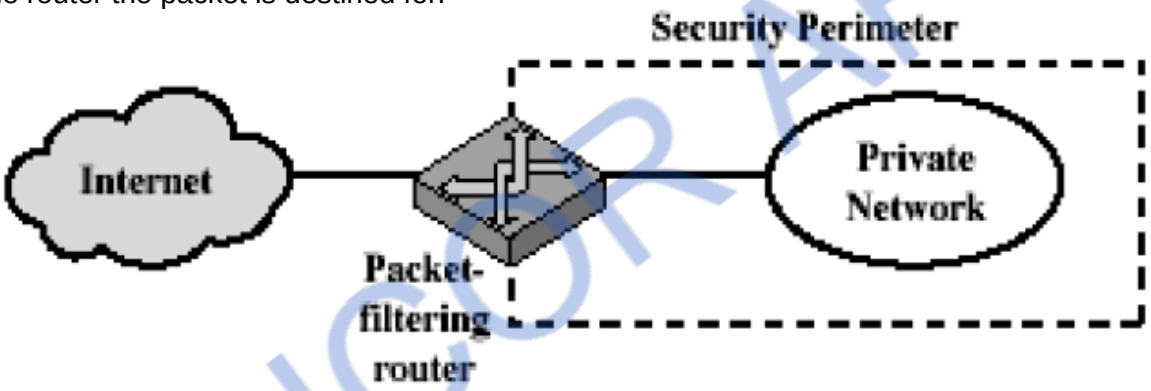
There are 3 common types of firewalls.

- Packet filters
- Application-level gateways
- Circuit-level gateways

Packet Filtering Router

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

- **Source IP address** – IP address of the system that originated the IP packet.
- **Destinations 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.



Packet Filtering Router

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.

Advantages of packet filter router

- Simple
- Transparent to users
- Very fast

Weakness of packet filter firewalls

- Packet filter firewalls do not examine upper-layer data; They cannot prevent attacks that employ application specific vulnerabilities or functions.
- As limited information is 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 on Packet Filtering Routers

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

- **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.

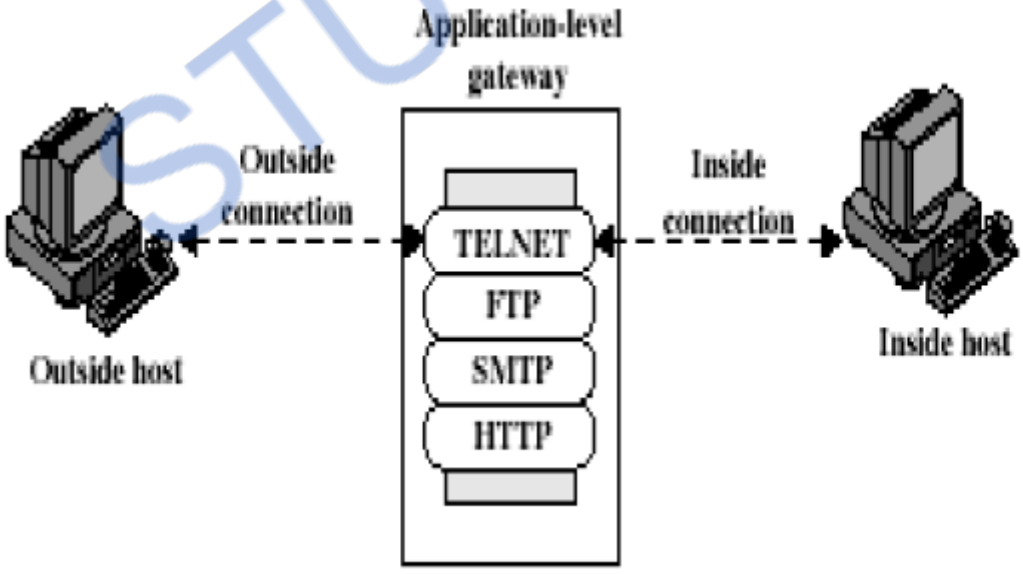
Countermeasure: to discard all packets where the protocol type is TCP and the IP fragment offset is equal to 1.

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.



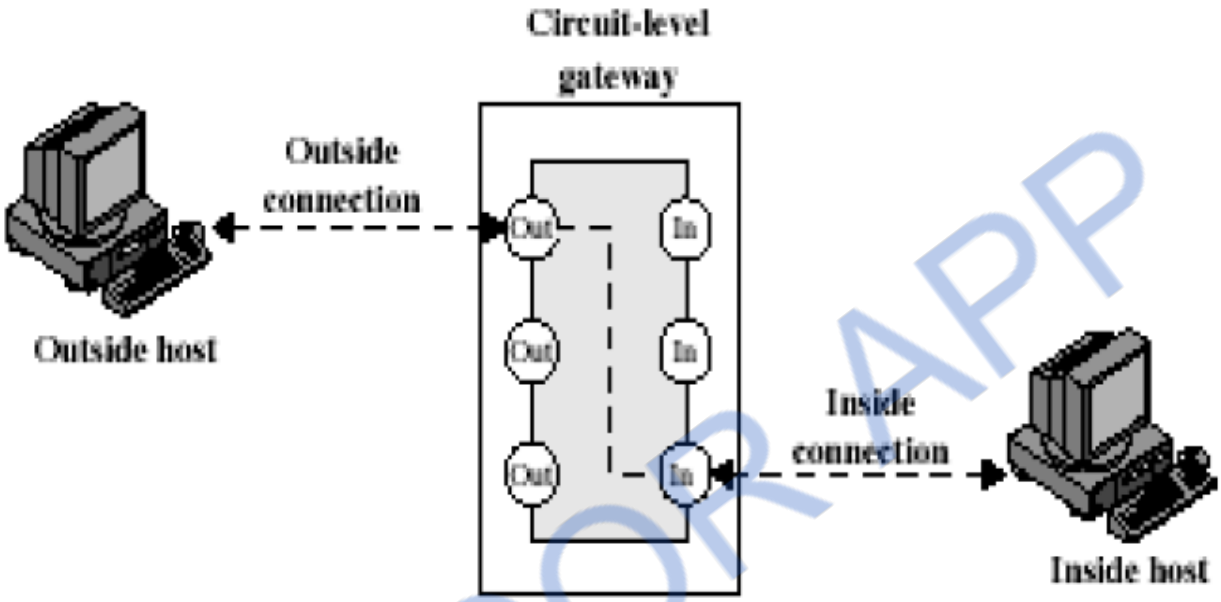
Application Level Gateway

Circuit Level Gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host.

Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.



Circuit Level Gateway