CS4101PC: INFORMATION SECURITY

IV-I:CSE								
Course Code	**Category**	**Hours/Week**			**Credits**	**Max Marks**		
CS4101PC	**Core**	**L**	**T**	**P**	**C**	**CIE**	**SEE**	**Total**
		3	**0**	**0**	**3**	**25**	**75**	**100**
Contact Classes:45	**Tutorial classes:15**	**Practical classes: Nill**				**Total Classes:60**		
Prerequisites								

Course Objectives:

- Explain the objectives of information security
- Explain the importance and application of each of confidentiality, integrity, authentication and availability
- Understand various cryptographic algorithms.
- Understand the basic categories of threats to computers and networks
- Describe public-key cryptosystem.
- Describe the enhancements made to IPv4 by IPSec
- Understand Intrusions and intrusion detection
- Discuss the fundamental ideas of public-key cryptography.
- Generate and distribute a PGP key pair and use the PGP package to send an encrypted e-mail message.
- Discuss Web security and Firewalls

Course Outcomes:

- Studentwillbeabletounderstandbasiccryptographicalgorithms,messageandwebauthenticationand security issues.
- Abilitytoidentifyinformationsystemrequirementsforbothofthemsuchasclientandserver.
- Abilitytounderstandthecurrentlegalissuestowardsinformationsecurity.

INFORMATION SECURITY (CS4101PC)

UNIT-I

SecurityConcepts: Introduction, Theneedforsecurity, Securityapproaches, Principlesofsecurity, Types of Security attacks, Security services, Security Mechanisms, A model for Network Security **CryptographyConceptsandTechniques:** Introduction, plaintextandciphertext, substitutiontechniques, transpositiontechniques, encryptionanddecryption, symmetrican d asymmetric key cryptography, steganography, key range and key size, possible types of attacks.

UNIT-II

SymmetrickeyCiphers: BlockCipherprinciples, DES, AES, Blowfish, RC5, IDEA, Block cipheroperation, Streamciphers, RC4.

AsymmetrickeyCiphers: Principlesofpublickeycryptosystems, RSAalgorithm, El gamalCryptography, Diffie-HellmanKey Exchange, Knapsack Algorithm.

UNIT-III

CryptographicHashFunctions: MessageAuthentication, SecureHashAlgorithm(SHA-512), **Message authentication codes:** Authentication requirements, HMAC, CMAC, Digital signatures, Elgamal Digital Signature Scheme.

Key Management and Distribution: Symmetric Key Distribution Using Symmetric & AsymmetricEncryption, DistributionofPublicKeys, Kerberos, X.509AuthenticationServi ce, Public–KeyInfrastructure

UNIT-IV

Transport-level Security: Web security considerations, Secure Socket Layer and Transport Layer Security, HTTPS, Secure Shell (SSH)

WirelessNetworkSecurity:

WirelessSecurity, MobileDeviceSecurity, IEEE802.11WirelessLAN, IEEE802.11iWirel essLAN Security

INFORMATION SECURITY (CS4101PC)

UNIT-V-

MailSecurity: PrettyGoodPrivacy,S/MIME**IPSecurity:**IPSecurityoverview,IPSecurity architecture, Authentication Header, Encapsulating security payload, Combining security associations, Internet Key Exchange

Case Studies on Cryptography and security : Secure MultipartyCalculation,VirtualElections,SinglesignOn,SecureInter-branchPaymentTransactions,CrosssiteScripting Vulnerability.

TEXTBOOKS:

1. Cryptography and Network Security-Principles and Practice: WilliamStallings,PearsonEducation,6th Edition
2. Cryptography and Network Security:AtulKahate,McGrawHill,3rd Edition

REFERENCEBOOKS:

1. CryptographyandNetworkSecurity:CKShyamala,NHarini,DrTRPadmanabhan, WileyIndia,1st Edition.
2. Cryptography and Network Security:ForouzanMukhopadhyay,McGrawHill,3rd Edition
3. Information Security, Principles, and Practice: Mark Stamp, Wiley India.
4. Principles of Computer Security: WM. Arthur Conklin ,Greg White, TMH
5. Introduction to Network Security: Neal Krawetz, CENGAGE Learning
6. Network Security and Cryptography :Bernard Menezes, CENGAGE Learning

INFORMATION SECURITY (CS4101PC)

LECTURE NOTES

UNIT-1

Security Concepts : Introduction, The need for security, Security approaches, Principles of security, Types of Security attacks,Security services, Security Mechanisms, A model for Network Security

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

1). INTRODUCTION TO INFORMATION SECURITY

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

Some examples are:

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

Network Security has been affected by two major developments over the last several decades. First one is introduction of computers into organizations and the second one being introduction of distributed systems and the use of networks and communication facilities for carrying data between users & computers. These two developments lead to 'computer security' and 'network security', where the computer security deals with collection of tools designed to protect data and to thwart hackers. Network security measures are needed to protect data during transmission. But keep in mind that, it is the information and our ability to access that information that we are really trying to protect and not the computers and networks.

2). NEED OF INFORMATION SECURITY

We need information security because there are threats.

Threat: A threat is an object, person, or other entity that represents a constant danger to an asset The 2007 CSI survey.

- 494 computer security practitioners
- 46% suffered security incidents
- 29% reported to law enforcement

- Average annual loss$350,424
- 1/5 suffered _targeted attack‘
- The source of the greatest financial losses?
- Most prevalent security problem
- Insider abuse of network access
- Email

Threat Categories:

- Acts of human error or failure
- Compromises to intellectual property
- Deliberate acts of espionage or trespass
- Deliberate acts of information extortion
- Deliberate acts of sabotage or vandalism
- Deliberate acts of theft
- Deliberate software attack
- Forces of nature
- Deviations in quality of service
- Technical hardware failures or errors
- Technological obsolesce

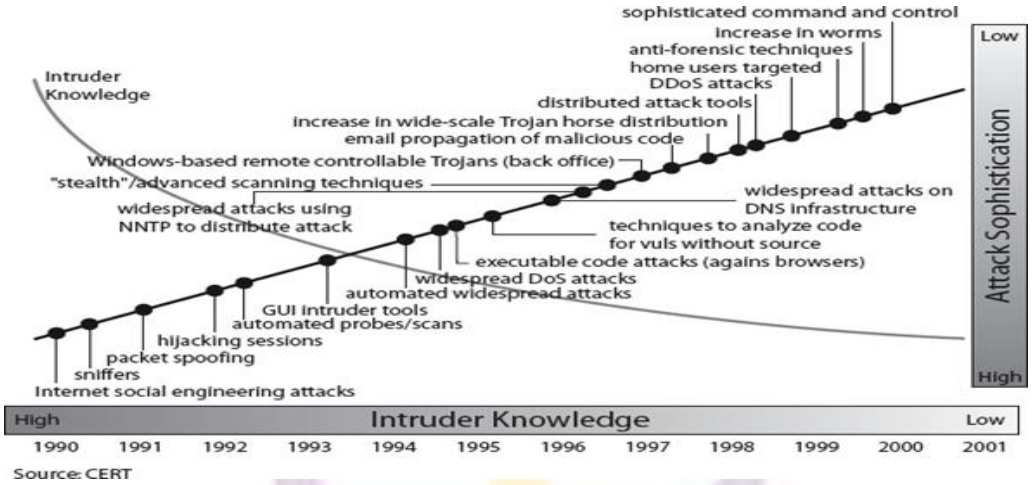
3). SECURITY APPROACHES:

Definitions

Computer Security - generic name for the collection of tools designed toprotect data and to thwart hackers

Network Security - measures to protect data during their transmission

Internet Security - measures to protect data during their transmission over acollection of interconnected networks our focus is on **Internet Security** which consists of measures to deter, prevent, detect, and correct security violations that involve the transmission & storage of information



4.) PRINCIPLES OF SECURITY:

The CIA triad refers to an information security model made up of the three main components: confidentiality, integrity and availability. Each component represents a fundamental objective of information security.

ASPECTS OF SECURITY:

Consider the three aspects of information security:

- Security attack
- Security mechanism
- Security service

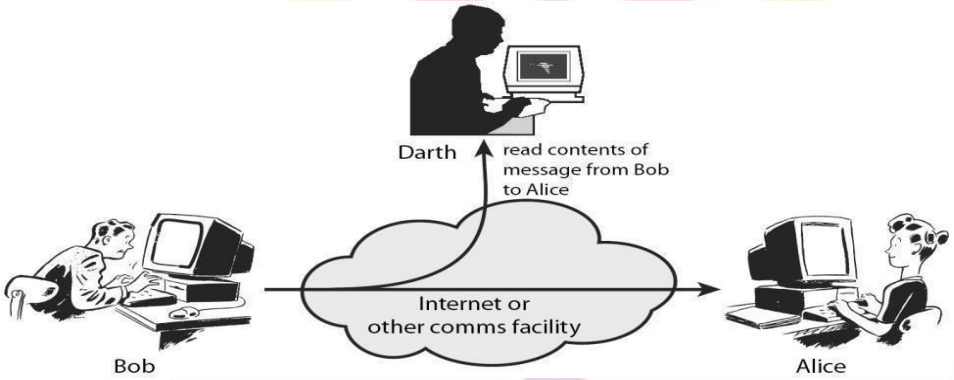
5). TYPES OF SECURITY ATTACKS

Any action that compromises the security of information owned by an organization.
Information security is about how to prevent attacks, or failing that, to detect attacks on

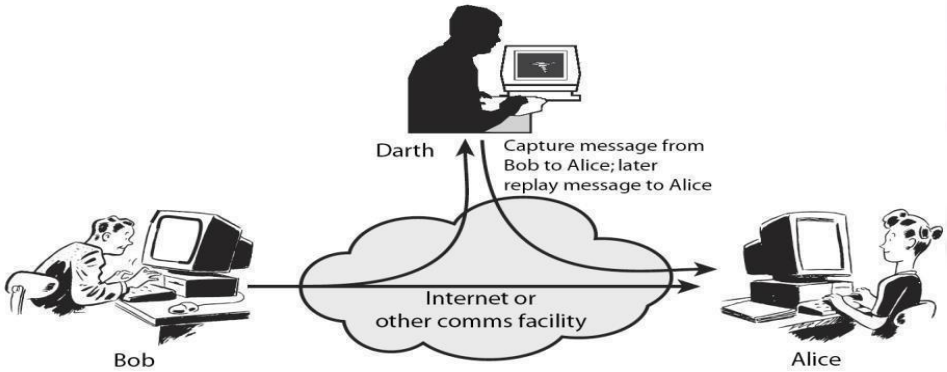
information-based systems.
Often threats & attacks used to mean same thing have a wide range of attacks can focus on generic types of attacks.

- Passive
- Active

Passive Attack:



Active Attack:



INTERRUPTION

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

Examples:

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

INTERCEPTION

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

Examples:

- Eavesdropping
- Wire tapping to capture data in a network.
- Illicitly copying data or programs

MODIFICATION:

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

Examples:

- Changing data file.
- Altering a program and the contents of a message.

FABRICATION:

An unauthorized party inserts a counterfeit object into the system. Attack on authenticity also called impersonation.

Examples:

- Hackers gaining access to a personal email and sending messages.
- Insertion of records in data files.
- Insertion of spurious message in a network.

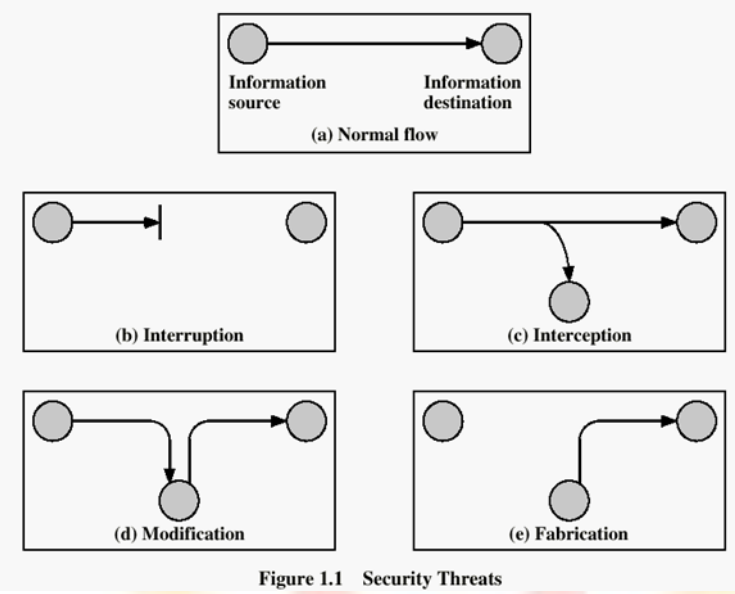


Figure 1.1 Security Threats

6).SECURITY SERVICES:

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

CONFIDENTIALIY:

Confidentiality is the protection of transmitted data from passive attacks. It is used to prevent the disclosure of information to unauthorized individuals or systems. It has been defined as "ensuring that information is accessible only to those authorized to have access".The other aspect of confidentiality is the protection of traffic flow from analysis. **Ex:** A credit card number has to be secured during online transaction.

Authentication

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

Peer entity authentication: Verifies the identities of the peer entities involved in communication. Provides use at time of connection establishment and during data transmission. Provides confidence against a masquerade or a replay attack

Data origin authentication: Assumes the authenticity of source of data unit, but does not

provide protection against duplication or modification of data units. Supportsapplications like electronic mail, where no prior interactions take place betweencommunicating entities.

Integrity

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

Connection-Oriented Integrity Service: This service deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering or replays. Destruction of data is also covered here. Hence, it attends to both message stream modification and denial of service.

Connectionless-Oriented Integrity Service: It deals with individual messages regardless of larger context, providing protection against message modification only.

An integrity service can be applied with or without recovery. Because it is related to active attacks, major concern will be detection rather than prevention. If a violation is detected and the service reports it, either human intervention or automated recovery machines are required to recover.

Non-repudiation

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

Access Control

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

Availability

It is defined to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity. The availability can significantly be affected by a variety of attacks, some amenable to automated counter measures i.e authentication and encryption and others need some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

7). SECURITY MECHANISMS

According to X.800, the security mechanisms are divided into those implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. X.800 also differentiates reversible & irreversible encipherment

mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted, whereas irreversible encipherment include hash algorithms and message authentication codes used in digital signature and message authentication applications

Specific Security Mechanisms

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

Encipherment: It refers to the process of applying mathematical algorithms for converting data into a form that is not intelligible. This depends on algorithm usedand encryption keys.

Digital Signature: The appended data or a cryptographic transformation applied to any data unit allowing to prove the source and integrity of the data unit and protect against forgery.

Access Control: A variety of techniques used for enforcing access permissions to the system resources.

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

Authentication Exchange: A mechanism intended to ensure the identity of an entityby means of information exchange.

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

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

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

Pervasive Security Mechanisms

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

Trusted Functionality: That which is perceived to b correct with respect to some criteria

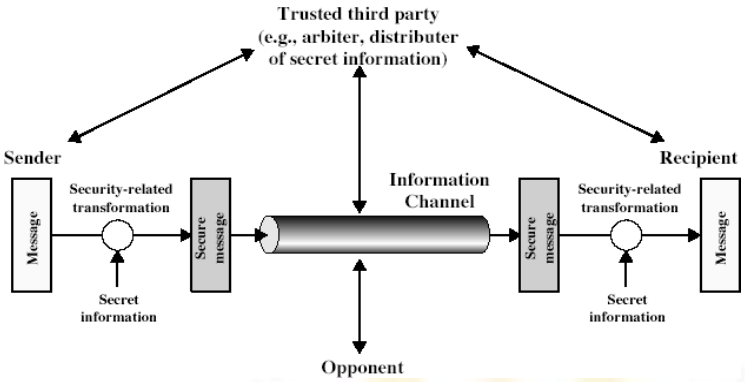
Security Level: The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection: It is the process of detecting all the events related to network security.

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

Security Recovery: It deals with requests from mechanisms, such as event handlingand management functions, and takes recovery actions.

8). MODEL FOR NETWORK SECURITY



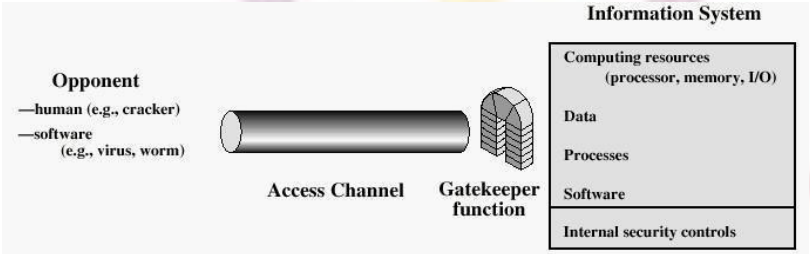


Figure 1.4 Network Access Security Model

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

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

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

A trusted third party may be needed to achieve secure transmission. It is responsible for distributing the secret information to the two parties, while keeping it away from any opponent. It also may be needed to settle disputes between the two parties regarding authenticity of a message transmission. The general model shows that there are four basic tasks in designing a particular security service:

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

algorithm should be such that an opponent cannot defeat its purpose

2. Generate the secret information to be used with the algorithm
3. Develop methods for the distribution and sharing of the secret information
4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service Various other threats to information system like unwanted access still exist.
5. The existence of hackers attempting to penetrate systems accessible over a network remains a concern. Another threat is placement of some logic in computer system affecting various applications and utility programs. This inserted code presents two kinds of threats.

Information access threats intercept or modify data on behalf of users who should not have access to that data

Service threats exploit service flaws in computers to inhibit use by legitimate users Viruses and worms are two examples of software attacks inserted into the system by means of a disk or also across the network. The security mechanisms needed to cope with unwanted access fall into two broad categories.

9).PLAIN TEXT & CIPHER TEXT

Some basic terminologies used

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

CRYPTOGRAPHY

Cryptographic systems are generally classified along 3 independent dimensions:

Type of operations used for transforming plain text to cipher text

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

The number of keys used

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

The way in which the plain text is processed

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

CRYPTANALYSIS

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst. **There are various types of cryptanalytic attacks** based on the amount of information known to the cryptanalyst.

Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.

Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.

Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

CLASSICAL ENCRYPTION TECHNIQUES

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

10). SUBSTITUTION TECHNIQUES

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

CAESAR CIPHER

The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet. e.g., plain text : pay more money Cipher text: SDB PRUH PRQHB

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

For each plaintext letter p, substitute the cipher text letter c such that C = E(p) = (p+3) mod 26

A shift may be any amount, so that general Caesar algorithm is C = E (p) = (p+k) mod 26

Where k takes on a value in the range 1 to 25.
The decryption algorithm is simply P = D(C) = (C-k) mod 26

MONOALPHABETIC CIPHERS

Here, Plaintext characters are substituted by a different alphabet stream of characters shifted to the right or left by n positions. When compared to the Caesar ciphers, these monoalphabetic ciphers are more secure as each letter of the ciphertext can be any permutation of the 26 alphabetic characters leading to 26! or greater than $4 \times 10_{26}$ possible keys. But it is still vulnerable to cryptanalysis, when a cryptanalyst is aware of the nature of the plaintext, he can find the regularities of the language. To overcome these attacks, multiple substitutions for a single letter are used. For example, a letter can be substituted by different numerical cipher symbols such as 17, 54, 69….. etc. Even this method is not completely secure as each letter in the plain text affects on letter in the ciphertext.Or, using a common key which substitutes every letter of the plain text.

The key *ABCDEFGHIIJ KLMNOPQRSTUVWXYZ QWERTYUIIOPAS DFGHJ KLZXCV BNM*

Would encrypt the message

II think therefore II am into

OZIIOFAZIITKTYGKTOQD

But any attacker would simply break the cipher by using frequency analysis by observing the number of times each letter occurs in the cipher text and then looking upon the English letter frequency table. So, substitution cipher is completely ruined by these attacks. Monoalphabetic ciphers are easy to break as they reflect the frequency of the original alphabet. A countermeasure is to provide substitutes, known as homophones for a single letter.

PLAYFAIR CIPHERS

It is the best known multiple –letter encryption cipher which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair Cipher is a digram substitution cipher offering a relatively weak method of encryption. It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians and Germans during World War II. This was because Playfair is reasonably fast to use and requires no special equipment. A typical scenario for Playfair use would be to protect important but non-critical secrets during actual combat. By the time the enemy cryptanalysts could break the message, the information was useless to them. It is based around a 5x5 matrix, a copy of which is held by both communicating parties, into which 25 of the 26 letters of the alphabet (normally either j and i are represented by the same letter or x is ignored) are placed in a random fashion. For example, the plain text is *ShiSherry loves Heath Ledger* and the agreed key is *sherry*. The matrix will be built according to the following rules.

- in pairs,
- without punctuation,
- All Js are replaced with Is.

SH IS HE RR YL OV ES HE AT HL ED GE R

- Double letters which occur in a pair must be divided by an X or a Z.
- E.g. LI TE RA LL Y LI TE RA LX LY

SH IS HE RX RY LO VE SH EA TH LE DG ER The alphabet square is prepared using, a 5*5 matrix, no repetition letters, no Js and key is written first followed by the remaining alphabets with no i and j.

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

For the generation of cipher text, there are three rules to be followed by each pair ofletters.

letters appear on the same row： replace them with the letters to their immediateright respectively

letters appear on the same column： replace them with the letters immediately below respectively

not on the same row or column： replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. Based on the above three rules, the cipher text obtained for the given plain text is

HE GH ER DR YS IQ WH HE SC OY KR AL RY

Another example which is simpler than the above one can be given as:

Here, key word is *playfair*. Plaintext is *Hellothere hellothere* becomes----------*he lx lo th er ex* . Applying the rules again, for each pair, If they are in the same row, replace eachwith the letter to its right (mod 5)

he KG

If they are in the same column, replace each with the letter below it (mod 5)

lo RV

Otherwise, replace each with letter we'd get if we swapped their column indices

lx YV

So the cipher text for the given plain text is **KG YV RV QM GI KU**

$$\begin{bmatrix} p & l & a & y & f \\ i & r & b & c & d \\ e & g & h & k & m \\ n & o & q & s & t \\ u & v & w & x & z \end{bmatrix}$$

To decrypt the message, just reverse the process. Shift up and left instead of down and right. Drop extra x's and locate any missing I's that should be j's. The message will be back into the original readable form. no longer used by military forces because of the advent of digital encryption devices. Playfair is now regarded as insecure for any purpose because modern hand-held computers could easily break the cipher within seconds.

HILL CIPHER

It is also a multi letter encryption cipher. It involves substitution of '*m*' ciphertext letters for '*m'* successive plaintext letters. For substitution purposes using '*m'* linear equations, each of the characters are assigned a numerical values i.e. a=0, b=1, c=2, d=3,…….z=25. For example if m=3, the system can be defined as: $\mathbf{c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26}$ $\mathbf{c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26}$ $\mathbf{c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26}$ If we represent in matrix form, the above statements as matrices and column vectors:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

Thus, C = KP *mod26,* where C= Column vectors of length 3 P= Column vectors of length 3 K= 3x3 encryption key matrix. For decryption process, inverse of matrix **K** i.e. $\mathbf{K^{-1}}$ is required which is defined by the equation $KK^{-1} = K^{-1}K = I$, where I is the identity matrix that contains only 0's and 1's as its elements. Plaintext is recovered by applying K^{-1} to the cipher text. It is expressed as$C = E_K(P) = KP \ mod26$ $P = D_K(C) = K^{-1}C \ mod26. = K^{-1}KP = IP=P$

Example: The plain text is I can't do it and the size of m is 3 and key K is chosen as following

```
I can't do it
8 2 0 13 19 3 14 8 19
```

$$\begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix}$$

The encryption process is carried out as follows

$$\begin{pmatrix} 4 \\ 14 \\ 12 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} 8 \\ 2 \\ 0 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} 19 \\ 12 \\ 14 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} 13 \\ 19 \\ 3 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} 18 \\ 21 \\ 9 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} 14 \\ 8 \\ 19 \end{pmatrix} \pmod{26}$$

So, the encrypted text will be given as → **EOM TMY SVJ**

The main advantages of hill cipher are given below: perfectly hides single-letter frequencies.

It Use of **3x3** Hill ciphers can perfectly hide both the single letter and two-letter frequency information.

Strong enough against the attacks made only on the cipher text.

But, it still can be easily broken if the attack is through a known plaintext.

POLYALPHABETIC CIPHERS

In order to make substitution ciphers more secure, more than one alphabet can be used. Such ciphers are called **polyalphabetic**, which means that the same letter of a message can be represented by different letters when encoded. Such a one-to-many correspondence makes the use of frequency analysis much more difficult in order to crack the code. We describe one such cipher named for *Blaise de Vigenere* a 16-th century Frenchman. The **Vigenere cipher** is a polyalphabetic cipher based on using successively shifted alphabets, a different shifted alphabet for each of the 26 English letters. The procedure is based on the tableau shown below and the use of a keyword.The letters of the keyword determine the shifted alphabets used in the encoding process.

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

For the message COMPUTING GIVES INSIGHT and keyword LUCKY we proceed by repeating the keyword as many times as needed above the message, as follows.

L	U	C	K	Y	L	U	C	K	Y	L	U	C	K	Y	L	U	C	K	Y	L
C	O	M	P	U	T	I	N	G	G	I	V	E	S	I	N	S	I	G	H	T

Encryption is simple: Given a key letter x and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled *x* and the column labeled *y*; so for L, the ciphertext letter would be N. So, the ciphertext for the given plaintext would be given as:

L	U	C	K	Y	L	U	C	K	Y	L	U	C	K	Y	L	U	C	K	Y	L				
C	O	M	P	U	T	I	N	G	G	I	V	E	S	I	N	S	I	G	H	T				<==MESSAGE
N	I	O	Z	S	E	C	P	Q	E	T	P	G	C	G	Y	M	K	Q	F	E				<==Encoded Message

Decryption is equally simple: The key letter again identifies the row and position of ciphertext letter in that row decides the column and the plaintext letter is at the top of that column. The strength of this cipher is that there are multiple ciphetext letters for each plaintext letter, one for each unique letter of the keyword and thereby making the letter frequency information is obscured. Still, breaking this cipher has been made possible because this reveals some mathematical principles that apply in cryptanalysis. To overcome the drawback of the periodic nature of the keyword, a new technique is proposed which is referred as an autokey system, in which a key word is concatenated with the plaintext itself to provide a running key. For ex In the above example, the key would be *luckycomputinggivesin* Still, this scheme is vulnerable to cryptanalysis as both the key and plaintext share the same frequency distribution of letters allowing a statistical technique to be applied. Thus, the ultimate defense against such a cryptanalysis is to choose a keyword that is as long as plaintext and has no statistical relationship to it. A new system which works on binary data rather than letters is given as

$\mathbf{C_i = p_i k_i}$ where, p_i = ith binary digit of plaintext k_i = ith binary digit of key C_i= ith binary digit of ciphertext = = exclusive-or operation. Because of the properties of XOR,decryption is done by performing the same bitwise operation.

$\mathbf{p_i = C_i k_i}$ A very long but, repeation key word is used making cryptanalysis difficult.

11). TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing

some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

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

Plaintext = meet at the school house

To encipher this message with a rail fence of depth 2,

We write the message as follows: m e a t e c o l o s e t t h s h o h u e

The encryptedmessage is MEATECOLOSETTHSHOHUE

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

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

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

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

12). ENCRYPTION AND DECRYPTION:

There are various reasons for using these processes, but a few important reasons are:

It helps protect confidential and sensitive data like passwords or other credentials.

- It helps maintain the integrity of users' data, as data won't constantly change and can be frequently verified.
- Helps to ensure that an attacker or third party doesn't alter the files or documents sent with the message.
- Prevents plagiarism and protects IP address.
- Helpful for proper network communication where an attacker can't access encrypted data.
- Helps protect PII/PHI data and securely explore the internet while maintaining compliant standards.

13). SYMMETRIC AND ASYMMETRIC KEY CRYPTOGRAPHY:

Symmetric Key Cryptography

In symmetric key cryptography, an individual key is used for both encryption and decryption. The sender needs the key to encrypt the plaintext and sends the cipher document to the receiver. The receiver used the similar key (or ruleset) to decrypt the message and recover the plaintext. Because an individual key is used for both functions, symmetric key cryptography is also known as symmetric encryption.

Symmetric key cryptography schemes are usually categorized such as stream ciphers or block ciphers. Stream ciphers works on a single bit (byte or computer word) at a time and execute some form of feedback structure so that the key is constantly changing.

Asymmetric cryptography

Asymmetric cryptography uses two keys for encryption and decryption. It depends on the technique of public and private keys. A public key, which is interchanged between higher than one user. Data is decrypted by a private key, which is not transformed. It is slower but more secure. The public key used in this encryption technique is applicable to everyone, but the private key used in it is not revealed

In asymmetric encryption, a message that is encrypted utilizing a public key can be decrypted by a private key, while if the message is encrypted by a private key can be decrypted by utilizing the public key. Asymmetric encryption is broadly used in dayto- day communication channels, particularly on the internet.

14). STEGANOGRAPHY

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods ofcryptography render the message unintelligible to outsiders by various transformations of the text. A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message. e.g., (i) the sequence of first letters of each word of the overall message spells out the real (hidden) message. (ii) Subset of the words of the overall message is used to convey the hidden message. Various other techniques have been used historically, some of them are

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

Drawbacks of Steganography

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

15). KEY RANGE AND KEY SIZE, POSSIBLE TYPES OF ATTACKS:

The concept of key range and key-size are related to each other. Key Range is total number of keys from smallest to largest available key. An attacker usually is armed with the knowledge of the

cryptographic algorithm and the encrypted message, so only the actual key value remains the challenge for the attacker.

• If the key is found, the attacker can get original plaintext message. In the brute force attack, every possible key in the key-range is tried, until we get the right key.

• In the best case, the right key is found in the first attempt, in the worst case, the key is found in the last attempt. On an average, the right key is found after trying half of the possible keys in the key-range. Therefore by expanding the key range to a large extent, longer it will take for an attacker to find the key using brute-force attack.

• The concept of key range leads to the principle of key size. The strength of a cryptographic key is measured with the key size

• Key size is measured in bits and is represented using binary number system. Thus if the key range from 0 to 8, then the key size is 3 bits or in other words we can say if the size is bits then the key range is 0 to 256. Key size may be varying, depending upon the applications and the cryptographic algorithm being used, it can be 40 bits, 56 bits, 128 bits & so on. In order to protect the cipher-text against the brute-force attack, the key-size should be such that the attacker can not crack it within a specified amount of time.

From a practical viewpoint, a 40-bit key takes about 3 hours to crack, however a 41-bit key would take 6 hours and 42-bit key would take 12 hours & so on. This means every additional bit doubles the amount of time required to crack the key. We can assume that 128 bit key is quite safe, considering the capabilities of today's computers.

Descriptive Questions:

(a) 2 Marks Questions

1. Define security attack, security mechanism and security services.

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

Security mechanism: a mechanism that is designed to detect, prevent or recover from a securityattack.

Security services: a service that enhances the security of the data processing systems and theinformation transfers of an organization.

2. Mention the different types of security services.

- Authentication
- Confidentiality
- Data integrity
- Non repudiation
- Access control
- Availability

3. Define passive attack and active attack.

Passive attacks are in the nature of eavesdropping, or monitoring of transmissions. Thetypes of passive attack are

- Release of message content
- Traffic analysis

Active attacks involve some modification of data stream or creation of afalse stream. The types of active attack are

- Masquerade
- Replay
- Modification
- Denial of service

4. Define the following terms:

Plaintext: the original message to be transmitted.

Cipher text: the coded (encrypted) message or the scrambled message. Encryption / Enciphering : process of converting plain text to cipher text. Decryption/ Deciphering: process of converting cipher text to plain text.

5. What are the two basic functions used in encryption algorithms?

The two basic functions used in encryption algorithms are

- Substitution
- Transposition

6. Define Threat and attack.

Threat is a possible danger that might exploit a vulnerability to breach security and thus causepossible harm.

Attack is any attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset

7. What are the two approaches to attacking a cipher?

The two approaches to attack a cipher are:

1. Cryptanalysis
2. Brute-force attack

8. Define Brute-force attack.

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

9. What is Modification of messages

Modification of messages simply means that some portion of a legitimate message is altered,or that messages are delayed or reordered, to produce an unauthorized effect.

10. What is masquerade?

A **masquerade** takes place when one entity pretends to be a different entity. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

11. What is Reply?

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

12. Define Denial of service.

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

13. Define the two basic building blocks of encryption techniques.

- Substitution technique – it is one in which the letters of the plaintext are replaced by otherletters or by numbers or symbols.
- Transposition technique – it is one which performs some sort of permutation on theplaintext letters.

14. List out the components of encryption algorithm.

- Plaintext
- Encryption algorithm
- Secret key
- Cipher text
- Decryption algorithm

15.Mention few mono-alphabetic and poly-alphabetic ciphers.

Mono-alphabetic ciphers: - playfair cipher, hill cipher, Caesar cipher Poly-alphabetic ciphers: - vigenere cipher, one time pad cipher

16. What is steganography? Mention few techniques in it.

Steganography is a technique for hiding the original message. Some of the related techniques are

- Character marking
- Invisible ink
- Pin punctures
- Typewriter correction ribbon

17. Mention the functions involved in simplified DES.

- Initial permutation
- A complex function F_k with a key k1
- Switching
- A complex function F_k with a key k2

- Inverse permutation

18. Define stream cipher and block cipher.

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.

A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal block

19. Compare stream cipher with block cipher with example.

Stream cipher: Processes the input stream continuously and producing one element at a time. Example: Caeser cipher.

Block cipher: Processes the input one block of elements at a time producing an output block for eachinput block. Example: DES

20. What are the operations used in AES?

- Substitute bytes
- ShiftRows
- MixColumns
- AddRoundKey

21. Convert the Given Text "CRYPTOGRAPHY" into cipher text using Rail fenceTechnique.

In rail fence technique the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

C Y T G A H R P O R P Y

The cipher text is CYTGAH RPORPY.

22. What are the attacks that can be performed in the networks?

- Disclosure
- Traffic analysis
- Masquerade
- Content modification
- Sequence modification
- Timing modification
- Source repudiation
- Destination repudiation

23. What primitive operations are used in RC4

- Key expansion
- Encryption
- Decryption

(b) 10 Marks Questions

1. Explain in detail about OSI security architecture.

2. A) Explain the steps involved in RC4.
   B) Discuss different block cipher modes of operation

3. A) How NIST defines computer security? Give examples of recent computer security attacks which you know.
   B) AES consists of four functions in three layers. Which of the functions are primarily for confusion and which are primarily for diffusion? Which of the layers are for confusion and whichare for diffusion? Justify your answers. Or Explain in detail about AES

4. Draw a matrix that shows the relationship between security mechanisms and attacks.

5. A) Write short notes on security mechanisms.
   B) Explain in detail about the steps involved in DES.

6. Explain the following substitution techniques in network security.
   A) Caesar cipher B) Play fair cipher c) Polyalphabetic cipherOr
Explain about substitution ciphers in detail with an example.

7. A) what is meant by security attack? Explain various types of security attacks.
   B) What are the goals of security? Explain in detail about security Services?

8. Explain classical encryption techniques (Steps involved in each encryption technique like Caesar cipher, playfair cipher, hill cipher, vigenere cipher, one time pad cipher, rail fence, etc)

9. A) Explain about steganography, transposition cipher.
   B) Explain any three substitution ciphers with an example

UNIT-2

Symmetric key Ciphers: Block Cipher principles, DES, AES, Blowfish, RC5, IDEA, Block cipher operation, Stream ciphers, RC4.
Asymmetric key Ciphers: Principles of public key cryptosystems, RSA algorithm, Elgamal Cryptography, Diffie- Hellman Key Exchange, Knapsack Algorithm.

1). Block Cipher Principles

Block ciphers are built in the Feistel cipher structure. Block cipher has a specific number of rounds and keys for generating ciphertext. For defining the complexity level of an algorithm few design principles are to be considered.

These are explained as following below:

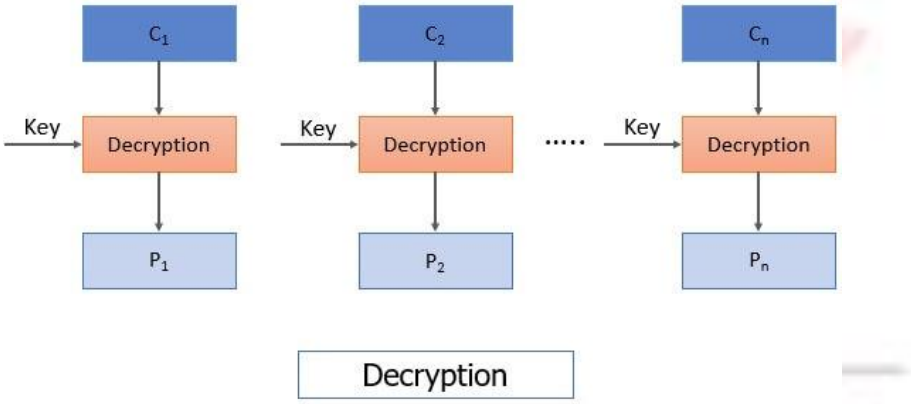
1. Number of Rounds –
   The number of Rounds is regularly considered in design criteria, it just reflects the number of

rounds to be suitable for an algorithm to make it more complex, in DES we have 16 rounds ensuring it to be more secure while in AES we have 10 rounds which makes it more secure.

2. Design of function F –
   The core part of the Feistel Block cipher structure is the Round Function. The complexity of cryptanalysis can be derived from the Round function i.e. the increasing level of complexity for the round function would be greatly contributing to an increase in complexity.

To increase the complexity of the round function, the avalanche effect is also included in the round function, as the change of a single bit in plain text would produce a mischievous output due to the presence of avalanche effect.

3. Key schedule algorithm –
   In Feistel Block cipher structure, each round would generate a sub-key for increasing the complexity of cryptanalysis. The Avalanche effect makes it more complex in deriving sub-key. Decryption must be done very carefully to get the actual output as the avalanche effect is present in it.



CONVENTIONAL ENCRYPTION PRINCIPLES

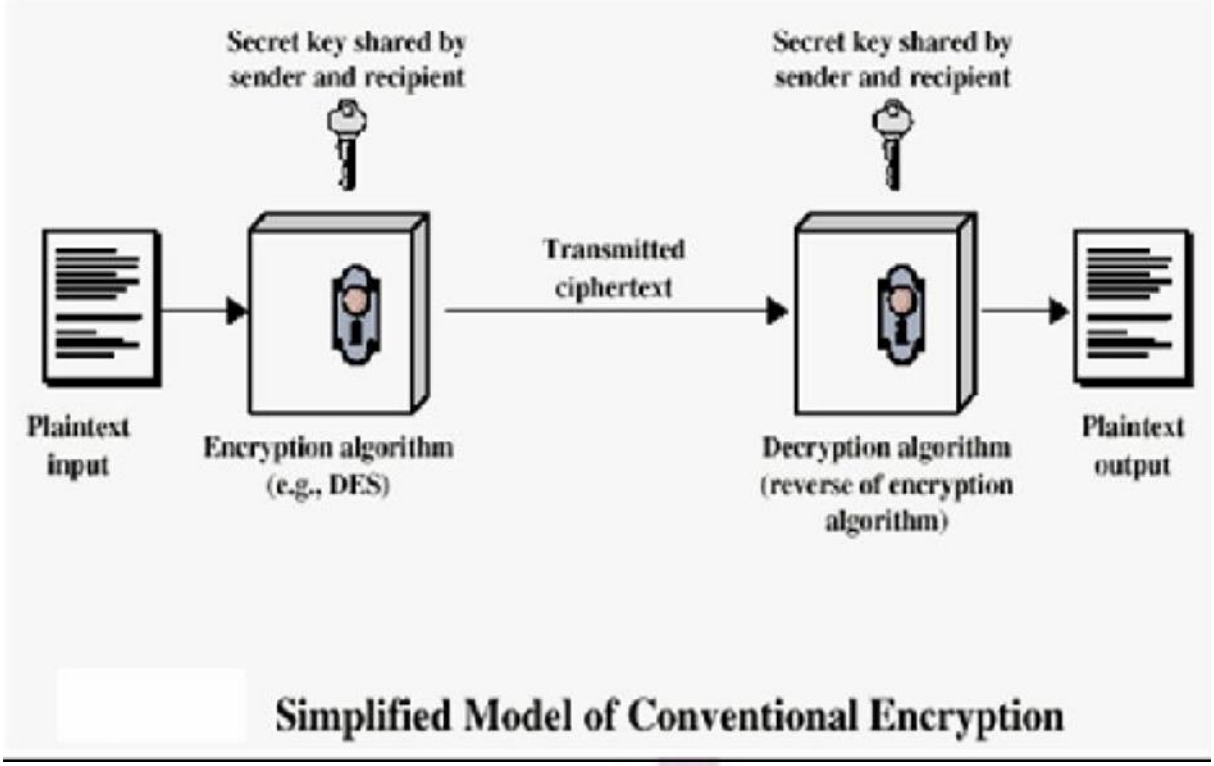
A Conventional/Symmetric encryption scheme has five ingredients

1. ***Plain Text***: This is the original message or data which is fed into the algorithm as input.

2. ***Encryption Algorithm***: This encryption algorithm performs various substitutions and transformations on the plain text.
3. ***Secret Key***: The key is another input to the algorithm. The substitutions andtransformations performed by algorithm depend on the key.
4. ***Cipher Text*:** This is the scrambled (unreadable) message which is output of the encryption algorithm. This cipher text is dependent on plaintext and secret key. For a

given plaintext, two different keys produce two different cipher texts.

5. ***Decryption Algorithm***: This is the reverse of encryption algorithm. It takes the ciphertext and secret key as inputs and outputs the plain text.



Simplified Model of Conventional Encryption

The important point is that the security of conventional encryption depends on the secrecy of the key, not the secrecy of the algorithm i.e. it is not necessary to keep the algorithm secret, but only the key is to be kept secret. This feature that algorithm neednot be kept secret made it feasible for wide spread use and enabled manufacturers develop low cost chip implementation of data encryption algorithms. With the use of conventional algorithm, the principal security problem is maintaining the secrecy of thekey.

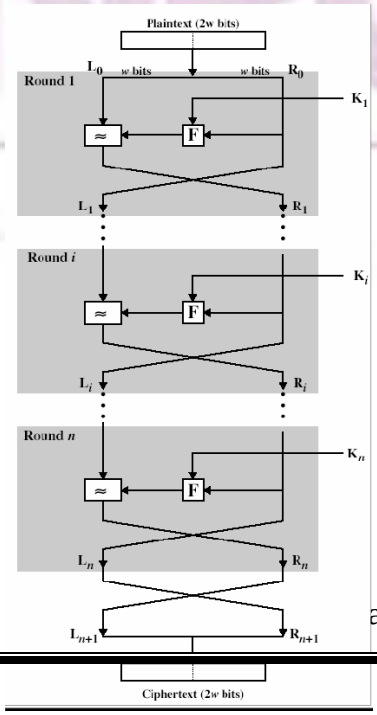
FEISTEL CIPHER STRUCTURE

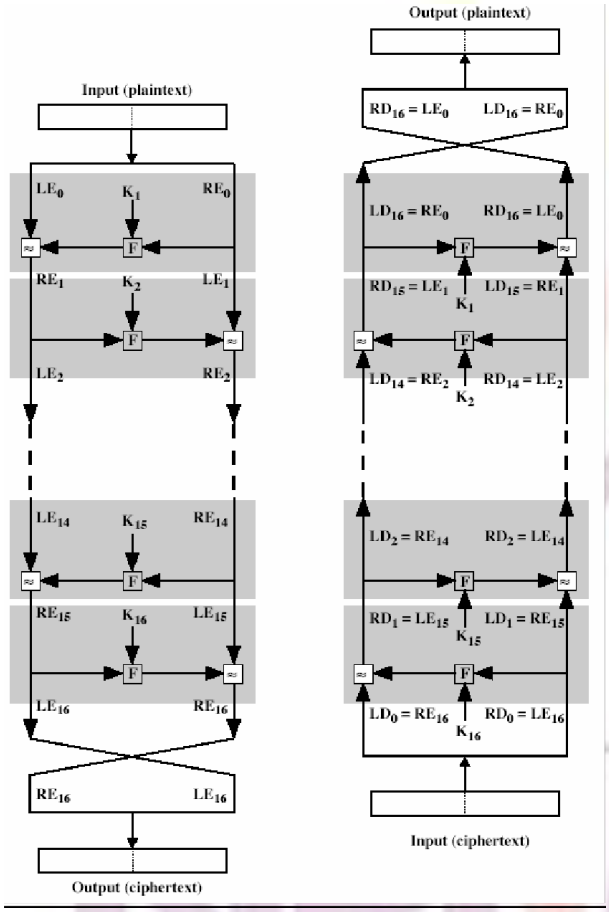
The input to the encryption algorithm are a plaintext block of length 2w bits and a key K. the plaintext block is divided into two halves L_0 and R_0. The two halves of the data pass through „n" rounds of processing and then combine to produce the ciphertext block. Each round „i" has inputs Li-1 and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

All rounds have the same structure. A substitution is performed on the left halfof the data

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

- **Block size** - Increasing size improves security, but slows cipher
- **Key size** - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **Number of rounds** - Increasing number improves security, but slows cipher
- **Subkey generation** - Greater complexity can make analysis harder, but slows cipher
- **Round function** - Greater complexity can make analysis harder, but slows cipher
- **Fast software en/decryption & ease of analysis** - are more recent concerns for practical use and testing





The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey k_i in reverse order. i.e., k_n in the first round, k_{n-1} in second round and so on. For clarity, we use the notation LE_i and RE_i for data traveling through the decryption algorithm. The diagram below indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the valueswapped.., $RE_i \| LE_i$ (or) equivalently $RD_{16-i} \| LD_{16-i}$

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

process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process.

First consider the encryption process, $LE_{16} = RE_{15}$
RE16 = LE15(+) F (RE15, K16)On the decryption side, $LD_1 = RD_0 = LE_{16} = RE_{15}$ $RD_1 = LD_0 \oplus F(RD_0, K_{16}) = RE_{16}\ F(RE_{15}, K_{16}) = [LE_{15}\ F(RE_{15}, K_{16})]\ F(RE_{15}, K_{16}) = LE_{15}$

Therefore, $LD_1 = RE_{15}$ $RD_1 = LE_{15}$ In general, for the i_{th} iteration of the encryption algorithm, $LE_i = RE_{i-1}$ $RE_i = LE_{i-1}\ F(RE_{i-1}, K_i)$
Finally, the output of the last round of the decryption process is $RE_0 \| LE_0$. A 32-bit swap recovers the original plaintext.

DEFINITIONS

Encryption: Converting a text into code or cipher.
Converting computer data and messages into something, incomprehensible use a key, so that only a holder of the matching key can reconvert them.

Conventional or Symmetric or Secret Key or Single Key encryption:

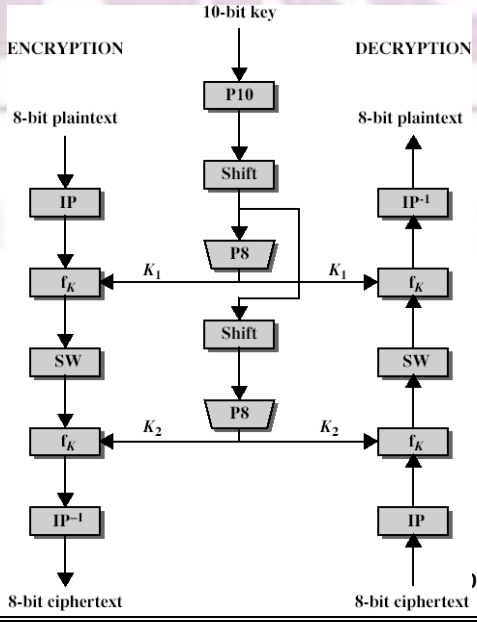
Uses the same key for encryption & decryption.
Public Key encryption: Uses different keys for encryption & decryption

Conventional Encryption Principles

- An encryption scheme has five ingredients:

1. Plaintext – Original message or data.
2. Encryption algorithm – performs substitutions & transformations on plaintext.
3. Secret Key – exact substitutions & transformations depend on this
4. Ciphertext - output ie scrambled input.
5. Decryption algorithm - converts ciphertext back to plaintext.

SIMPLIFIED DATA ENCRYPTION STANDARD (S-DES)



The figure above illustrates the overall structure of the simplified DES. The S- DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output. The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.

The encryption algorithm involves five functions:

- an initial permutation (IP)
- a complex function labeled f_k, which involves both permutation and substitution operations and depends on a key input
- a simple permutation function that switches (SW) the two halves of the data
- the function f_k again
- a permutation function that is the inverse of the initial permutation

The function f_k takes as input not only the data passing through the encryption algorithm, but also an 8-bit key. Here a 10-bit key is used from which two 8-bit subkeys are generated. The key is first subjected to a permutation (P10). Then a shift operation is performed. The output of the shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first subkey (K1). The output of the shift operation also feeds into another shift and another instance of P8 to produce the second subkey (K2).

The encryption algorithm can be expressed as a composition composition$_1$ of functions: IP^{-1} o fK2 o SW o fk1 o IP Which can also be written as

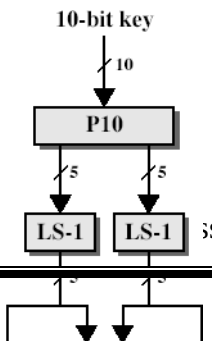
Ciphertext = IP^{-1} (f_{K2} (SW (f_{k1} (IP (plaintext)))))

K1 = P8 (Shift (P10 (Key)))

K2 = P8 (Shift (shift (P10 (Key))))

Decryption can be shown as

Plaintext = IP^{-1} (f_{K1} (SW (f_{k2} (IP (ciphertext)))))



S-DES depends on the use of a 10-bit key shared between sender and receiver.
From this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm. First, permute the key in the following fashion. Let the 10-bit key be designated as (k1, K2, k3, k4, k5, k6, k7, k8, k9, k10). Then the permutation P10 is defined as:
P10 (k1, K2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, K2, k7, k4, k10 10, k1, k9, k8, k6)
P10 can be concisely defined by the display:

P10									
3	5	2	7	4	10	1	9	8	6

This table is read from left to right; each position in the table gives the identity of the input bit that produces the output bit in that position. So the first output bit is bit 3 of the input; the second output bit is bit 5 of the input, and so on. For example, the key (1010000010) is permuted to (10000 01100). Next, perform a circular left shift (LS- 1), or rotation, separately on the first five bits and the second five bits. In our example, the result is (00001 11000). Next we apply P8, which picks out and permutes 8 of the 10 bits according to the following rule:

P8							
6	3	7	4	8	5	10	9

The result is subkey 1 (K1). In our example, this yields (10100100). We then go back to the pair of 5-bit strings produced by the two LS-1 functions and performs a circular left shift of 2 bit positions on each string. In our example, the value (00001 11000) becomes (00100 00011). Finally, P8 is applied again to produce K2. In our example, the result is (01000011).

S-DES encryption

Encryption involves the sequential application of five functions.

Initial and Final Permutations The input to the algorithm is an 8-bit block of plaintext, which we first permute using the IP function:

IP							
2	6	3	1	4	8	5	7

This retains all 8 bits of the plaintext but mixes them up. Consider the plaintext to be 11110011.

Permuted output = 10111101

At the end of the algorithm, the inverse permutation is used:

IP –1							
4	1	3	5	7	2	8	6

The Function f_k

The most complex component of S-DES is the function f_k, which consists of a combination of permutation and substitution functions. The functions can be expressed as follows. Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8- bit input to f K, and let F be a mapping (not necessarily one to one) from 4-bit strings to 4-bit strings.

Then we let $f_k(L, R) = (L (+) F(R, SK), R)$

Where SK is a subkey and (+) is the bit-by-bit exclusive-OR function.

e.g., permuted output = 1011 1101 and suppose F (1101, SK) = (1110) for some key SK.

Then f K(10111101) = 10111110, 1101 = 01011101

We now describe the mapping F. The input is a 4-bit number (n1 n2 n3 n4). The first operation is an expansion/permutation operation:

E/P							
4	1	2	3	2	3	4	1

R= 1101 E/P output = 11101011 It is clearer to depict the result in this fashion:

n_4	n_1	n_2	n_3
n_2	n_3	n_4	n_1

The 8-bit subkey K1 = (k11, k12 12, k13 13, k14 14, k15 15, k16 16, k17 17, k18) is added to this value using exclusive-OR:

$n_4 \oplus k_{11}$	$n_1 \oplus k_{12}$	$n_2 \oplus k_{13}$	$n_3 \oplus k_{14}$
$n_2 \oplus k_{15}$	$n_3 \oplus k_{16}$	$n_4 \oplus k_{17}$	$n_1 \oplus k_{18}$

Let us rename these 8 bits:

$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	$p_{0,3}$
$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$

The first 4 bits (first row of the preceding matrix) are fed into the s-box to produce a 2-bit output, and the remaining 4 bits are fed into s1 to produce another 2-bit output.These two boxes are defined as follows:

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

The S-boxes operate as follows. The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the S-box. The entry in that row and column, in base 2, is the 2-bit output. For example, if (p0,0 p0,3) =) (00) and (p0,1 p0,2) = (10), then the output is from row 0, column 2 of S0, which is 3, or (11) in) binary. Similarly, (p1,0 p1,3) and (p1,1 p1,2) are used to index into a row and column of S1 to produce anadditional 2 bits. Next, the 4 bits produced by S0 and S1 undergo a further permutation as follows:

P4			
2	4	3	1

The output of P4 is the output of the function F.

The Switch Function The function f K only alters the leftmost 4 bits of the input. The switch function (SW) interchanges the left and right 4 bits so that the second instance of f Koperates on a different 4 bits. In this second instance, the E/P, S0, S1, and P4 functions are the same. The key input is K2. Finally apply inverse permutation to get the ciphertext

2). DATA ENCRYPTION STANDARD (DES)

The main standard for encrypting data was a symmetric algorithm known as the Data Encryption Standard (DES). However, this has now been replaced by a new standard known as the Advanced Encryption Standard (AES) which we will look at later. DES is a 64 bit block cipher which means that it encrypts data 64 bits at a time. This is contrasted to a stream cipher in which only one bit at a time (or sometimes small groups of bits such as a byte) is encrypted. DES was the result of a research project set up by International Business Machines (IBM) Corporation in the late1960's which resulted in a cipher known as LUCIFER. In the early 1970's it was decided to commercialize LUCIFER and a number of significant changes were introduced. IBM was not the only one involved in these changes as they sought technical advice from the National Security Agency (NSA)

(other outside consultants were involved but it is likely that the NSA were the major contributors from a technical point of view). The altered version of LUCIFER was put forward as a proposal for the new national encryption standard requested by the National Bureau of Standards (NBS)3 . It was finally adopted in 1977 as the Data Encryption Standard DES (FIPS PUB 46). Some of the changes made to LUCIFER have been the subject of much controversy even to the present day. The most notable of these was the key size. LUCIFER used a key size of 128 bits however this was reduced to 56 bits for DES. Even though DES actually accepts a 64 bit key as input, the remaining eight bits are used for parity checking and have no effect on DES's security. Outsiders were convinced that the 56 bit key was an easy target for a brute force attack4 due to its extremely small size. The need for the parity checking scheme was also questioned without satisfying answers. Another controversial issue was that the S-boxes used were designed under classified conditions and no reasons for their particular design were ever given. This led people to assume that the NSA had introduced a "trapdoor" through which they could decrypt any data encrypted by DES even without knowledge of the key. One startling discovery was that the S-boxes appeared to be secure against an attack known as Differential Cryptanalysis which was only publicly discovered by Biham and Shamir in 1990. This suggests that the NSA were aware of this attack in 1977; 13 years earlier! In fact the DES designers claimed that the reason they never made the design specifications for the S-boxes available was that they knew about a number of attacksthat weren't public knowledge at the time and they didn't want them leaking - this is quite a plausible claim as differential cryptanalysis has shown. However, despite all this controversy, in 1994 NIST reaffirmed DES for government use for a further five years for use in areas other than "classified". DES of course isn't the only symmetric cipher. There are many others, each with varying levels of complexity. Such ciphers include: IDEA, RC4, RC5, RC6 and the new Advanced Encryption Standard (AES). AES is an important algorithm and was originally meant to replace DES (and its more secure variant triple DES) as the standard algorithm for non-classified material. However as of 2003, AES with key sizes of 192 and 256 bits has been found to be secure enough to protect information up to top secret. Since its creation, AES had underdone intense scrutiny as one would expect for an algorithm that is to be used as the standard. To date it has withstood all attacks but the search is still on and it remains to be seen whether or not this will last. We will look at AES later in the course.

INNER WORKING OF DES

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. It consists of a number of rounds where each round contains bit- shuffling, non- linear substitutions (S-boxes) and exclusive OR operations. As with most encryption schemes, DES expects two inputs - the plaintext to be encrypted and the secret key. The manner in which the plaintext is accepted, and the key arrangement used for encryption and decryption, both determine the type of

cipher it is. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and only operates on 64 bit blocks of data at a time5 (be they plaintext or ciphertext). The key sizeused is 56 bits, however a 64 bit (or eight-byte) key is actually input. The least significant bit of each byte is either used for parity (odd for DES) or set arbitrarily and does not increase the security in any way. All blocks are numbered from left to right which makes the eight bit of each byte the paritybit.

Once a plain-text message is received to be encrypted, it is arranged into 64 bit blocks required for input. If the number of bits in the message is not evenly divisible by 64, then the last block will be padded. Multiple permutations and substitutions are incorporated throughout in order to increase the difficulty of performing a cryptanalysis on the cipher.

OVERALL STRUCTURE:

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

So in m
the lef
2. The
xt and

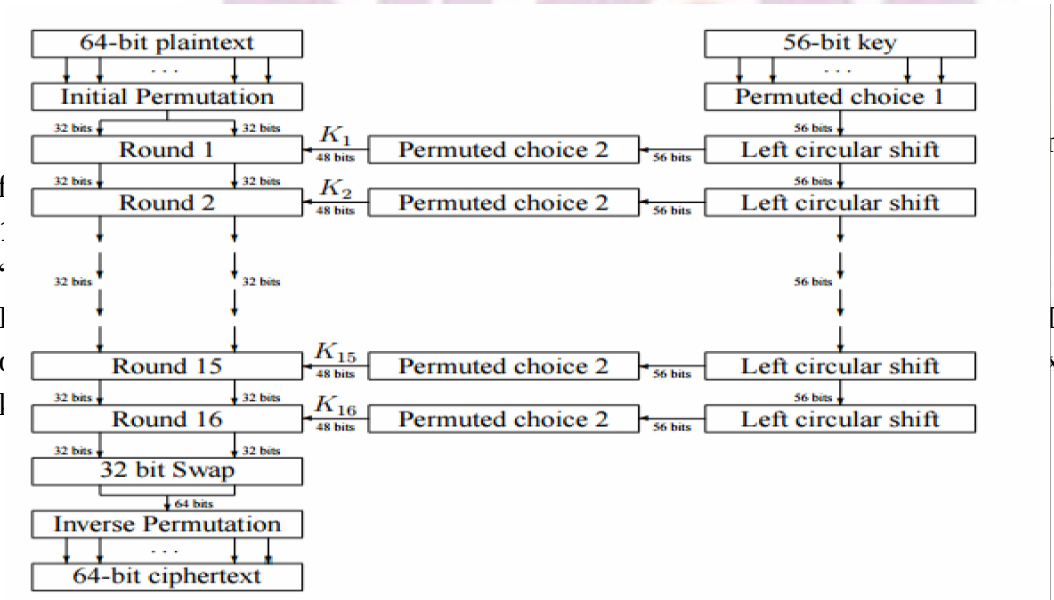


Figure : Flow Diagram of DES algorithm for encrypting data.

(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

(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

Table 2.1: Permutation tables used in DES.

As the figure shows the input to each round consists of the the Li , Ri pair and a 48 bit subkey which is a shifted and contracted version of the original 56 bit key. The use of the key can be seen in the right hand portion of figure 2.2: • Initially the key is passed through a permutation function (PC1 - defined in table 2.2) • For each of the 16 iterations, a subkey (Ki) is produced by a combination of a left circular shift and a permutation (PC2 - defined in table 2.2) which is the same for each iteration. However, the resulting subkey is different for each iteration because of repeated shifts.

(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

DETAILS OF INDIVIDUAL ROUNDS

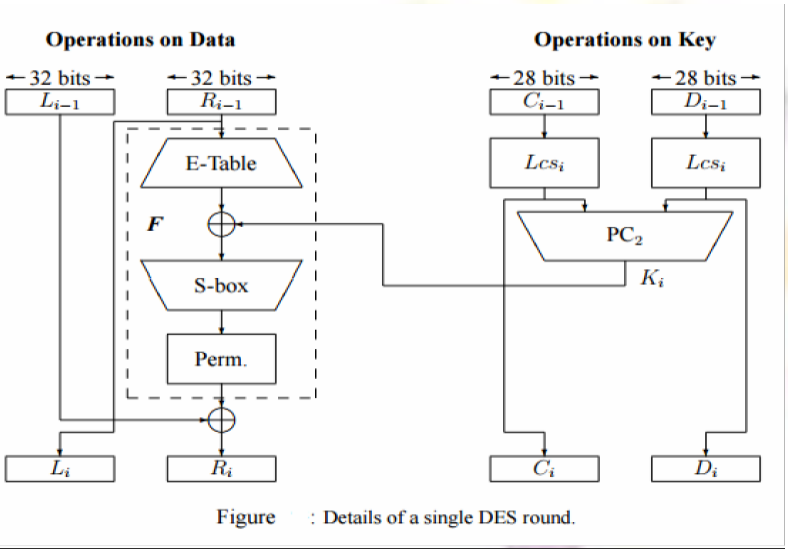


Figure : Details of a single DES round.

The main operations on the data are encompassed into what is referred to as the cipher function and is labeled F. This function accepts two different length inputs of 32 bits and 48 bits and outputs a single 32 bit number. Both the data and key are operated on in parallel, however the operations are quite different. The 56 bit key is split into two 28 bit halves Ci and Di (C and D being chosen so as not to be confused with L and R). The value of the key used in any round is simply a left cyclic shift and a permuted contraction of that used in the previous round.

Mathematically, this can be written as

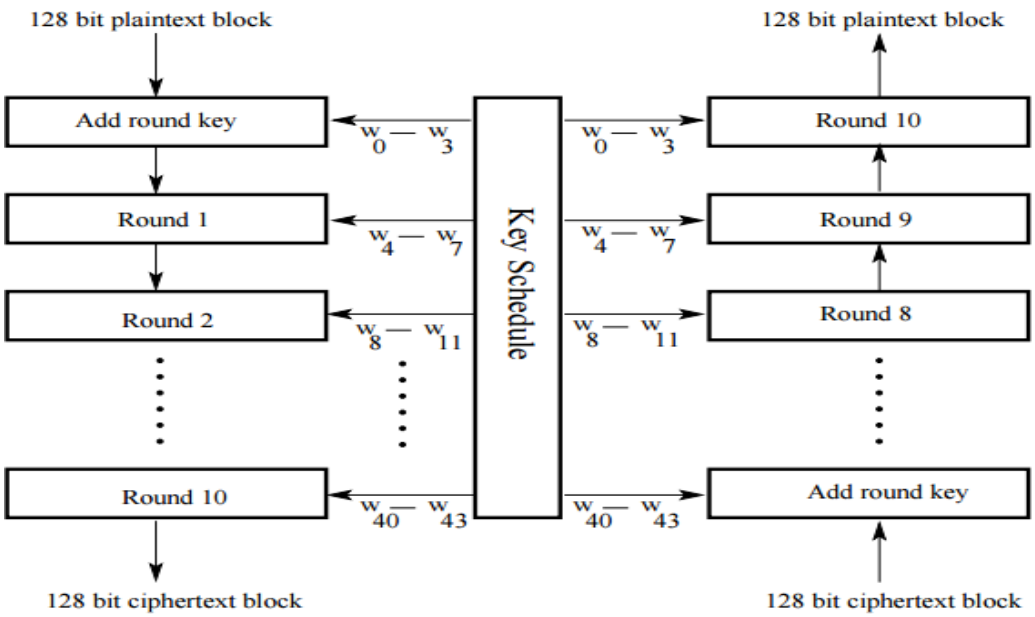
Ci=Lcsi(Ci-1)

Di=Lcsi(Di-1)

Ki=P C2(Ci,Di)

where Lcsi is the left cyclic shift for round i, Ci and Di are the outputs after the shifts, P C2(.) is a function which permutes and compresses a 56 bit number into a 48 bit number and Ki is the actual key used in round i. The number of shifts is either one or two and is determined by the round number i. For i = {1, 2, 9, 16} the number of shifts is one and for every other round it is two.

3). AES



AES Encryption AES Decryption

The algorithm begins with an Add round key stage followed by 9 rounds of fourstages and a tenth round of three stages.

This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of its counterpart in the encryption algorithm.

The four stages are as follows: 1. Substitute bytes 2. Shift rows 3. Mix Columns 4. AddRound Key

Substitute Bytes

• This stage (known as SubBytes) is simply a table lookup using a 16 × 16 matrix ofbyte values called an s-box.

• This matrix consists of all the possible combinations of an 8 bit sequence(28 = 16 ×16
= 256).

• However, the s-box is not just a random permutation of these values and there is a well defined method for creating the s-box tables.

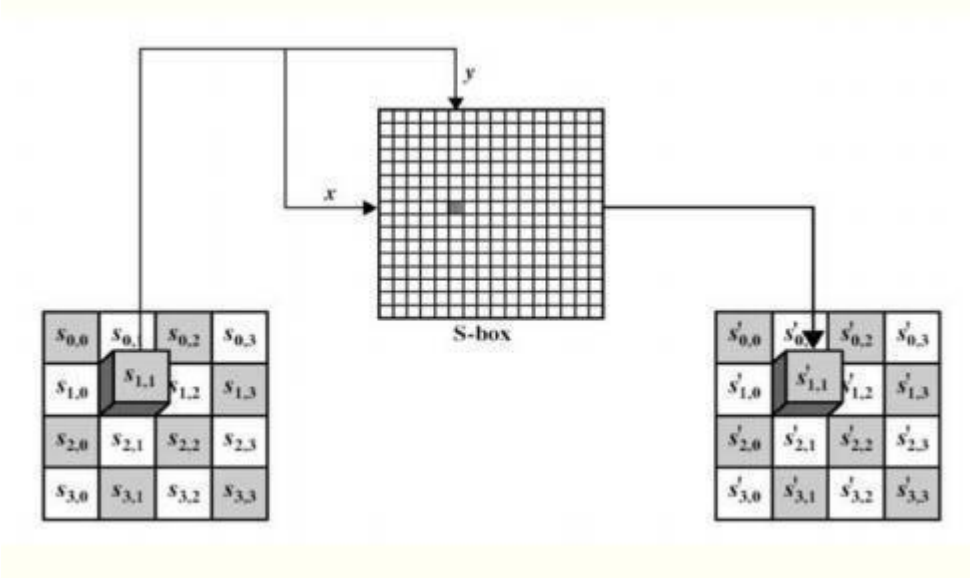
• The designers of Rijndael showed how this was done unlike the s-boxes in DES for which no rationale was given.Our concern will be how state is effected in eachround.

• For this particular round each byte is mapped into a new byte in the following way:

the leftmost nibble of the byte is used to specify a particular row of the s-box and the rightmost nibble specifies a column.

• For example, the byte {95} (curly brackets represent hex values in FIPS PUB 197) selects row 9 column 5 which turns out to contain the value {2A}.

This is then used to update the state matrix.

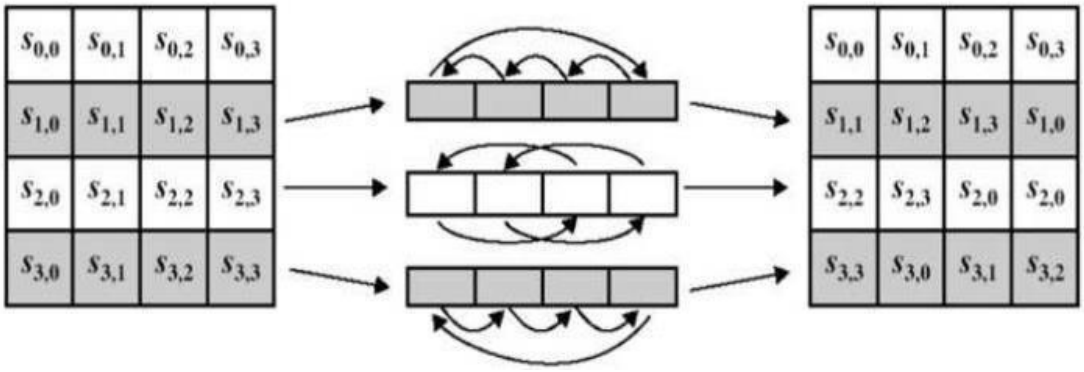


Shift Row Transformation

• This stage (known as ShiftRows) is shown in figure below.

• Simple permutation an nothing more.

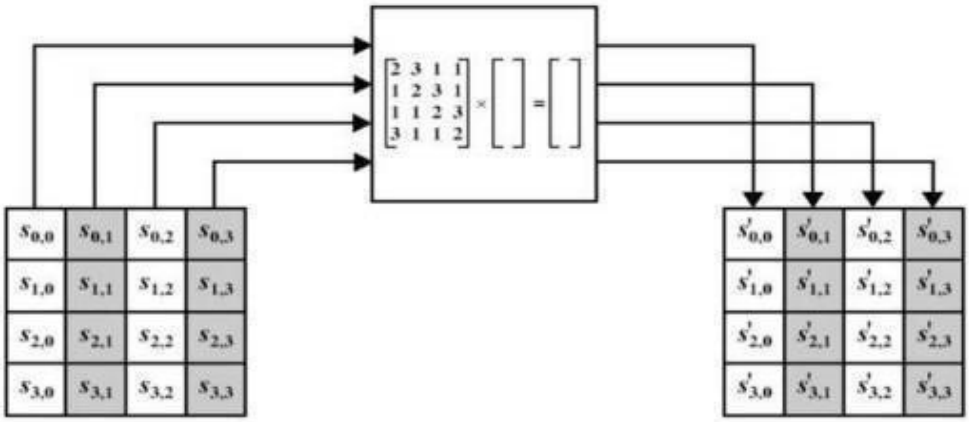
• It works as follow: – The first row of state is not altered. – The second row is shifted 1 bytes to the left in a circular manner. – The third row is shifted 2 bytes to the left ina circular manner. – The fourth row is shifted 3 bytes to the left in a circular manner.



MIX COLUMN TRANSFORMATION

- This stage (known as MixColumn) is basically a substitution
- Each column is operated on individually. Each byte of a column is mapped into a new value that is a function of all four bytes in thecolumn.
- The transformation can be determined by the following matrix multiplication on state
- Each element of the product matrix is the sum of products of elements of one row and one column.
- In this case the individual additions and multiplications are performed in GF(28).

The MixColumns transformation of a single column j ($0 \leq j \leq 3$) of state can be



ADD ROUND KEY TRANSFORMATION

- In this stage (known as AddRoundKey) 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.
- This transformation is as simple as possible which helps in efficiency but it also effects every bit of state.
- The AES key expansion algorithm takes as input a 4-word key and produces a linear array of 44 words. Each round uses 4 of these words as shown infigure.
- Each word contains 32 bytes which means each subkey is 128 bits long. Figure7show pseudocode for generating the expanded key from the actual key.

4). BLOWFISH ALGORITHM

- a symmetric block cipher designed by Bruce Schneier in1993/94
- characteristics
  - fast implementation on 32-bit CPUs
  - compact in use of memory
  - simple structure for analysis/implementation
  - variable security by varying key size
- has been implemented in various products

BLOWFISH KEY SCHEDULE

- uses a 32 to 448 bit key, 32-bit words stored in K-array K_j ,j from 1 to 14
- used to generate
  - 18 32-bit subkeys stored in P array, P_1 ….P_{18}
  - four 8x32 S-boxes stored in $S_{i,j}$, each with 256 32-bitentries

1. initialize P-array and then 4 S-boxes in order using the fractionalpart of pi P_1 (left most 32-bit), and so on,,, $S_{4,255.}$
2. XOR P-array with key-Array (32-bit blocks) and reuse as needed: assume we have up to k_{10} then P_{10} XOR K_{10},, P_{11} XOR K_1 … P_{18} XOR K_8

Encrypt 64-bit block of zeros, and use the result to update P_1 and$P_{2.}$

1. encrypting output form previous step using current P & S and replace P3 and P4. Then encrypting current output and use it to
2. update successive pairs of P.
3. After updating all P's (last :P_{17} P_{18}), start updating S values using the encrypted output from previous step.

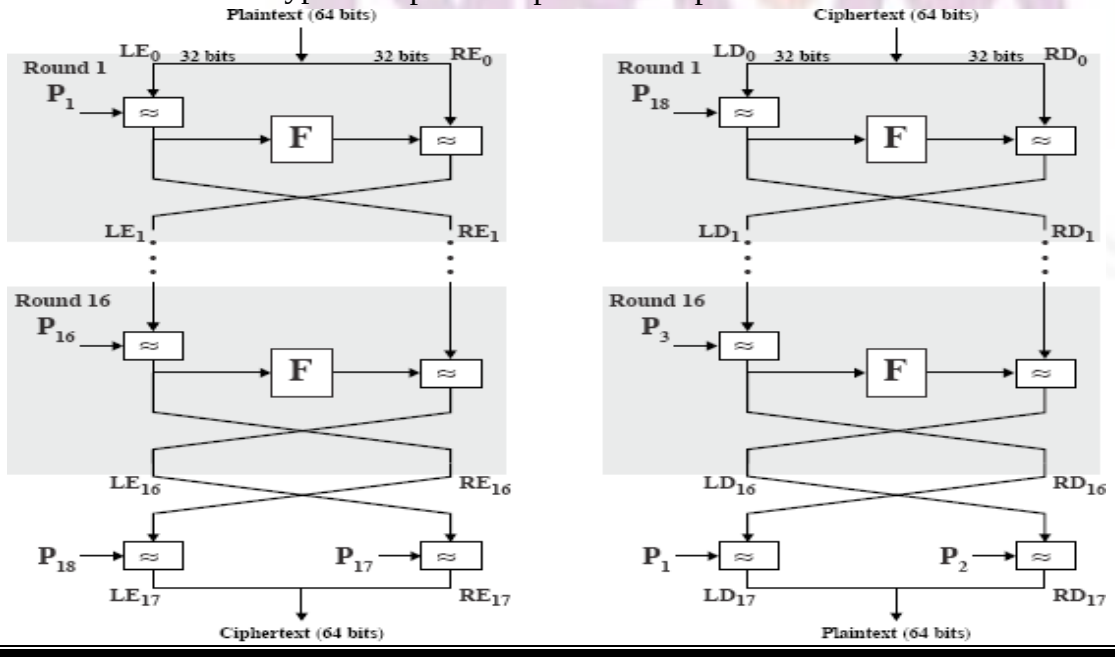


Figure 6.3 Blowfish Encryption and Decryption

- requires 521 encryptions, hence slow in re-keying
- Not suitable for limited-memory applications.

BLOWFISH ENCRYPTION

- uses two main operations: addition modulo 2^{32} , andXOR
- data is divided into two 32-bit halves L_0 &R_0

  for i = 1 to 16 do

  $R_i = L_{i-1}$ XOR P_i;

  $R_i = L_{i-1}$ XOR P_i; R_{i1}

  $L_{17}=R_{16}$X O R P_{18};

5). RC5 Encryption Algorithm

RC5 is a symmetric key block encryption algorithm designed by Ron Rivest in 1994. It is notable for being simple, fast (on account of using only primitive computer operations like XOR, shift, etc.) and consumes less memory.

Example:

Key : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Plain Text : 00000000 00000000

Cipher Text : EEDBA521 6D8F4B15

RC5 is a block cipher and addresses two word blocks at a time.
Depending on input plain text block size, number of rounds and key size, various instances of RC5 can be defined and each instance is denoted as RC5-w/r/b where w=word size in bits, r=number of rounds and b=key size in bytes.
Allowed values are:

Parameter	Possible Value
block/word size (bits)	16, 32, 64

Parameter	Possible Value
Number of Rounds	0 – 255
Key Size (bytes)	0 – 255

Note – Since at a time, RC5 uses 2 word blocks, the plain text block size can be 32, 64 or 128 bits.

Notation used in the algorithm:

Symbol	Operation
x <<< y	Cyclic left shift of x by y bits
+	Two's complement addition of words where addition is modulo
^	Bit wise Exclusive-OR

Step-1: Initialization of constants P and Q.
RC5 makes use of 2 magic constants P and Q whose value is defined by the word size w.

Word Size (bits)	P (Hexadecimal)	Q (Hexadecimal)
16	b7e1	9e37
32	b7e15163	9e3779b9
64	b7e151628aed2a6b	9e3779b97f4a7c15

For any other word size, P and Q can be determined as:

P = Odd((e-2))

Q = Odd((-2))

Here, Odd(x) is the odd integer nearest to x, e is the base of natural logarithms and is the golden ratio.

Step-2: Converting secret key K from bytes to words.
Secret key K of size b bytes is used to initialize array L consisting of c words where c = b/u, u = w/8 and w = word size used for that particular instance of RC5. For example, if we choose w=32 bits and Key k is of size 96 bytes then, u=32/8=4, c=b/u=96/4=24.
L is pre initialized to 0 value before adding secret key K to it.

for i=b-1 to 0

L[i/u] = (L[u/i] <<< 8) + K[i]

Step-3: Initializing sub-key S.
Sub-key S of size t=2(r+1) is initialized using magic constants P and Q.

S[0] = P

for i = 1 to 2(r+1)-1

S[i] = S[i-1] + Q)

Step-4: Sub-key mixing.
The RC5 encryption algorithm uses Sub key S. L is merely, a temporary array formed on the basis of user entered secret key.
Mix in user's secret key with S and L.

i = j = 0

A = B = 0

do 3 * max(t, c) times:

A = S[i] = (S[i] + A + B) <<< 3

B = L[j] = (L[j] + A + B) <<< (A + B)

i = (i + 1) % t

j = (j + 1) % c

Step-5: Encryption.
We divide the input plain text block into two registers A and B each of size w bits. After undergoing the encryption process the result of A and B together forms the cipher text block.
RC5 Encryption Algorithm:

1. One time initialization of plain text blocks A and B by adding S[0] and S[1] to A and B respectively. These operations are mod .
2. XOR A and B. A=A^B
3. Cyclic left shift new value of A by B bits.
4. Add S[2*i] to the output of previous step. This is the new value of A.
5. XOR B with new value of A and store in B.

6. Cyclic left shift new value of B by A bits.
7. Add S[2*i+1] to the output of previous step. This is the new value of B.
8. Repeat entire procedure (except one time initialization) r times.

A = A + S[0]

B = B + S[1]

for i = 1 to r do:

A = ((A ^ B) <<< B) + S[2 * i]

B = ((B ^ A) <<< A) + S[2 * i + 1]

return A, B

Alternatively, RC5 Decryption can be defined as:

for i = r down to 1 do:

B = ((B - S[2 * i + 1]) >>> A) ^ A

A = ((A - S[2 * i]) >>> B) ^ B

B = B - S[1]

A = A - S[0]

return A, B

6). Simplified International Data Encryption Algorithm (IDEA)

In cryptography, block ciphers are very important in the designing of many cryptographic algorithms and are widely used to encrypt the bulk of data in chunks. By chunks, it means that the cipher takes a fixed size of the plaintext in the encryption process and generates a fixed size ciphertext using a fixed-length key. An algorithm's strength is determined by its key length.

The Simplified **International Data Encryption Algorithm (IDEA)** is a **symmetric key block** cipher that:

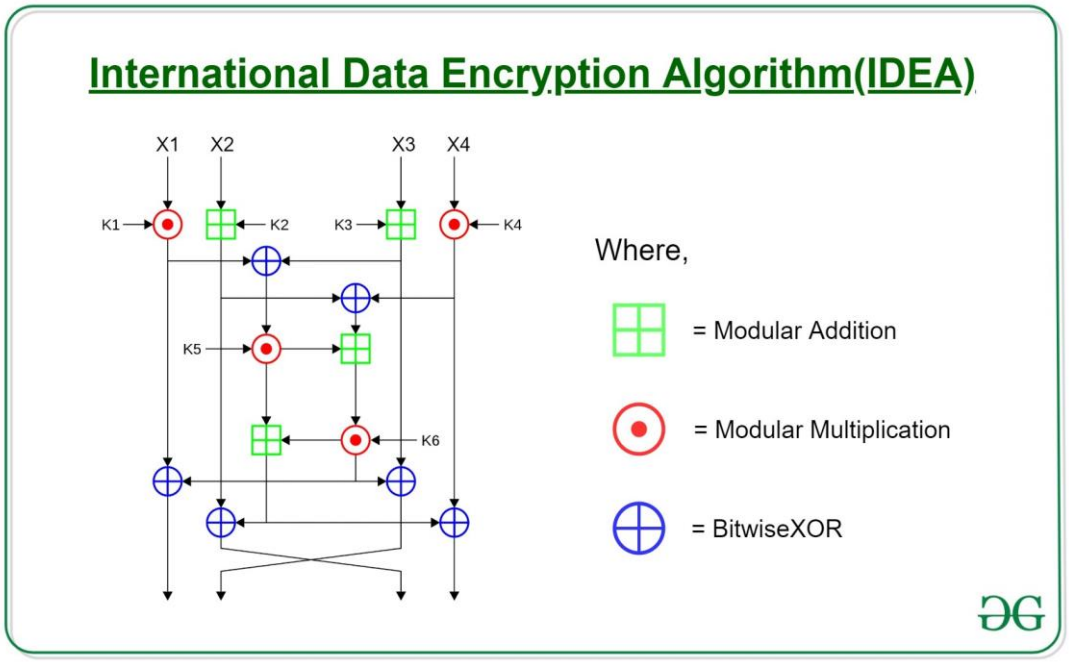
- uses a fixed-length plaintext of **16 bits** and
- encrypts them in **4 chunks of 4 bits** each
- to produce **16 bits ciphertext**.
- The length of the key used is **32 bits**.
- The key is also divided into 8 blocks of 4 bits each.

This algorithm involves a series of 4 identical complete rounds and 1 half-round. Each complete round involves a series of 14 steps that includes operations like:

- Bitwise XOR
- Addition modulo
- Multiplication modulo +1

After 4 complete rounds, the final "half-round" consists of only the first 4 out of the 14 steps previously used in the full rounds. To perform these rounds, each binary notation must be converted to its equivalent decimal notation, perform the operation and the result obtained should be converted back to the binary representation for the final result of that particular step.

Key Schedule: 6 subkeys of 4 bits out of the 8 subkeys are used in each complete round, while 4 are used in the half-round. So, 4.5 rounds require 28 subkeys. The given key, 'K', directly gives the first 8 subkeys. By rotating the main key left by 6 bits between each group of 8, further groups of 8 subkeys are created, implying less than one rotation per round for the key (3 rotations).



	K1	K2	K3	K4	K5	K6
Round 1	1101	1100	0110	1111	0011	1111
Round 2	0101	1001*	0001	1011	1100	1111
Round 3	1101	0110	0111	0111*	1111	0011

	K1	K2	K3	K4	K5	K6
Round 4	1111	0101	1001	1101	1100	0110*****
Round 4.5	1111	1101	0110	0111		

* denotes a shift of bits

Notations used in the 14 steps:

Symbol	Operation
*	Multiplication modulo +1
+	Addition modulo
^	Bitwise XOR

The 16-bit plaintext can be represented as **X1 || X2 || X3 || X4**, each of size 4 bits. The 32-bit key is broken into 8 subkeys denoted as K1 || K2 || K3 || K4 || K5 || K6 || K7 || K8, again of size 4 bits each. Each round of 14 steps uses the three algebraic operation-Addition modulo (2^4), Multiplication modulo (2^4)+1 and Bitwise XOR. The steps involved are as follows:

1. X1 * K1
2. X2 + K2
3. X3 + K3
4. X4 * K4
5. Step 1 ^ Step 3
6. Step 2 ^ Step 4
7. Step 5 * K5
8. Step 6 + Step 7
9. Step 8 * K6

10. Step 7 + Step 9
11. Step 1 ^ Step 9
12. Step 3 ^ Step 9
13. Step 2 ^ Step 10
14. Step 4 ^ Step 10

The input to the next round is Step 11 || Step 13 || Step 12 || Step 14, which becomes X1 || X2 || X3 || X4. This swap between 12 and 13 takes place after each complete round, except the last complete round (4th round), where the input to the final half round is Step 11 || Step 12 || Step 13 || Step 14.

After last complete round, the half-round is as follows:

1. X1 * K1
2. X2 + K2
3. X3 + K3
4. X4 * K4

The final output is obtained by concatenating the blocks.

Example:

Key: 1101 1100 0110 1111 0011 1111 0101 1001

Plaintext: 1001 1100 1010 1100

Ciphertext: 1011 1011 0100 1011

Explanation:
The explanation is only for 1st complete round (the remaining can be implemented similarly) and the last half round.

- **Round 1:**
    - From the plaintext: **X1 – 1001, X2 – 1100, X3 – 1010, X4 – 1100**
    - From the table above: **K1 – 1101, K2 – 1100, K3 – 0110, K4 – 1111, K5 – 0011, K6 – 1111**

(1001(9) * 1101(13))(mod 17) = 1111(15)

(1100(12) + 1100(12))(mod 16) = 1000(8)

(1010(10) + 0110(6))(mod 16) = 0000(0)

(1100(12) * 1111(15))(mod 17) = 1010(10)

(1111(15) ^ 0000(0)) = 1111(15)

(1000(8) ^ 1010(10)) = 0010(2)

(1111(15) * 0011(3))(mod 17) = 1011(11)

(0010(2) + 1011(11))(mod 16) = 1101(13)

(1101(13) * 1111(15))(mod 17) = 1000(8)

(1011(11) + 1000(8))(mod 16) = 0011(3)

(1000(8) ^ 1111(15)) = 0111(7)

(1000(8) ^ 0000(0)) = 1000(8)

(0011(3) ^ 1000(8)) = 1011(11)

(0011(3) ^ 1010(10)) = 1001(9)

-
- **Round 1 Output**: 0111 1011 1000 1001 *(Step 12 and Step 13 results are interchanged)*
- **Round 2:**
    - From Round 1 output: **X1 – 0111, X2 – 1011, X3 – 1000, X4 – 1001**
    - From the table above: **K1 – 0101, K2 – 1001, K3 – 0001, K4 – 1011, K5 – 1100, K6 – 1111**
    - **Round 2 Output**: 0110 0110 1110 1100 *(Step 12 and Step 13 results are interchanged)*
- **Round 3:**
    - From Round 2 Output: **X1 – 0110, X2 – 0110, X3 – 1110, X4 – 1100**
    - From the table above: **K1 – 1101, K2 – 0110, K3 – 0111, K4 – 0111, K5 – 1111, K6 – 0011**
    - **Round 3 Output**: 0100 1110 1011 0010 *(Step 12 and Step 13 results are interchanged)*
- **Round 4:**
    - From Round 3 Output: **X1 – 0100, X2 – 1110, X3 – 1011, X4 – 0010**
    - From the table above: **K1 – 1111, K2 – 0101, K3 – 1001, K4 – 1101, K5 – 1100, K6 – 0110**
    - **Round 4 Output**: 0011 1110 1110 0100 *(Step 12 and Step 13 results are interchanged)*
- **Round 4.5:**

- From Round 4 Output: **X1 – 0011, X2 – 1110, X3 – 1110, X4 – 0100**
- From the table above: **K1 – 1111, K2 – 1101, K3 – 0110, K4 – 0111**
- **Round 4.5 Output**: 1011 1011 0100 1011 *(Step 2 and Step 3 results are **not** interchanged)*
-

(0011(3) * 1111(15))(mod 17) = 1011(11)

(1110(14) + 1101(13))(mod 16) = 1011(11)

(1110(14) + 0110(6))(mod 16) = 0100(4)

(0100(4) * 0111(7))(mod 17) = 1011(11)

-
- **Final Ciphertext** is **1011 1011 0100 1011**

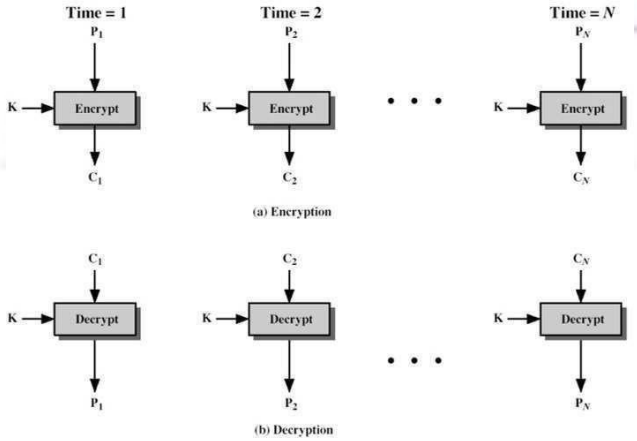
7). BLOCK CIPHER OPERATIONS

• Direct use of a block cipher is inadvisable
• Enemy can build up "code book" of plaintext/ciphertext equivalents
• Beyond that, direct use only works on messages that are a multipleof the cipher block size in length
• Solution: five standard Modes of Operation: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), OutputFeedback (OFB), and Counter(CTR).

Electronic Code Book

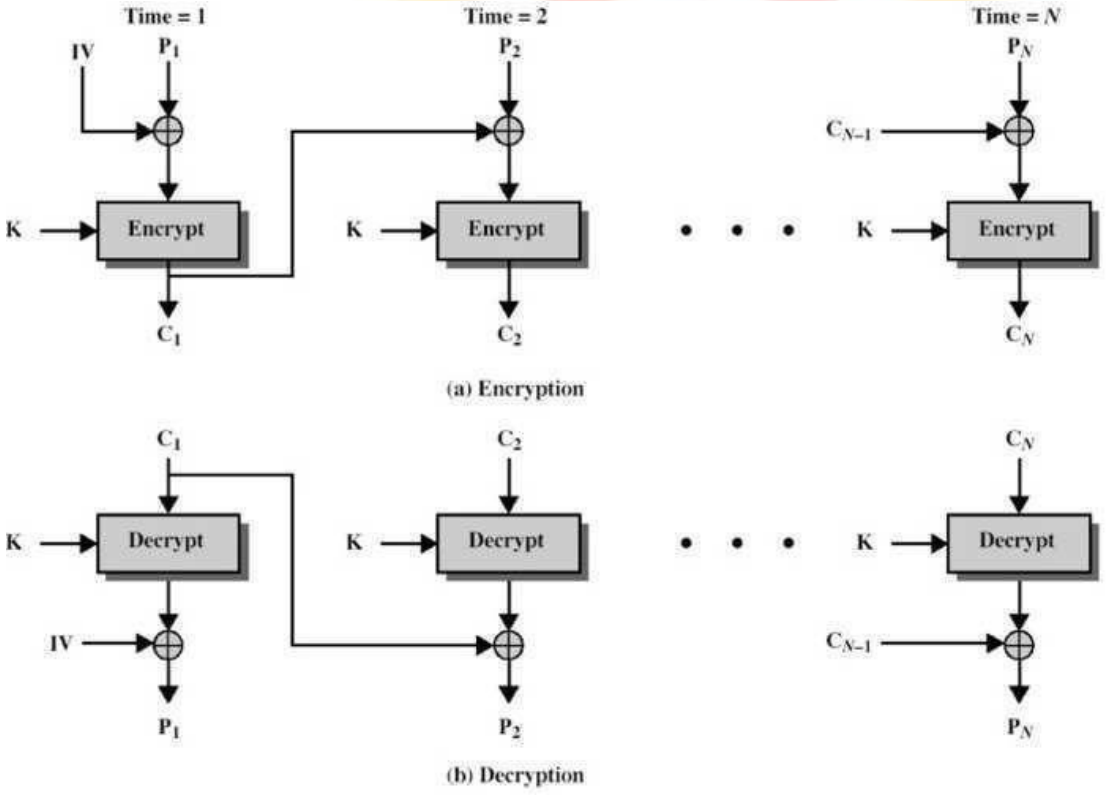
• Direct use of the block cipher
• Used primarily to transmit encrypted keys
• Very weak if used for general-purpose encryption; never use it for a file or amessage.
• Attacker can build up codebook; no semanticsecurity

• We write {P}k → C to denote "encryption of plaintext P with key k to produce ciphertext



CIPHER BLOCK CHAINING:

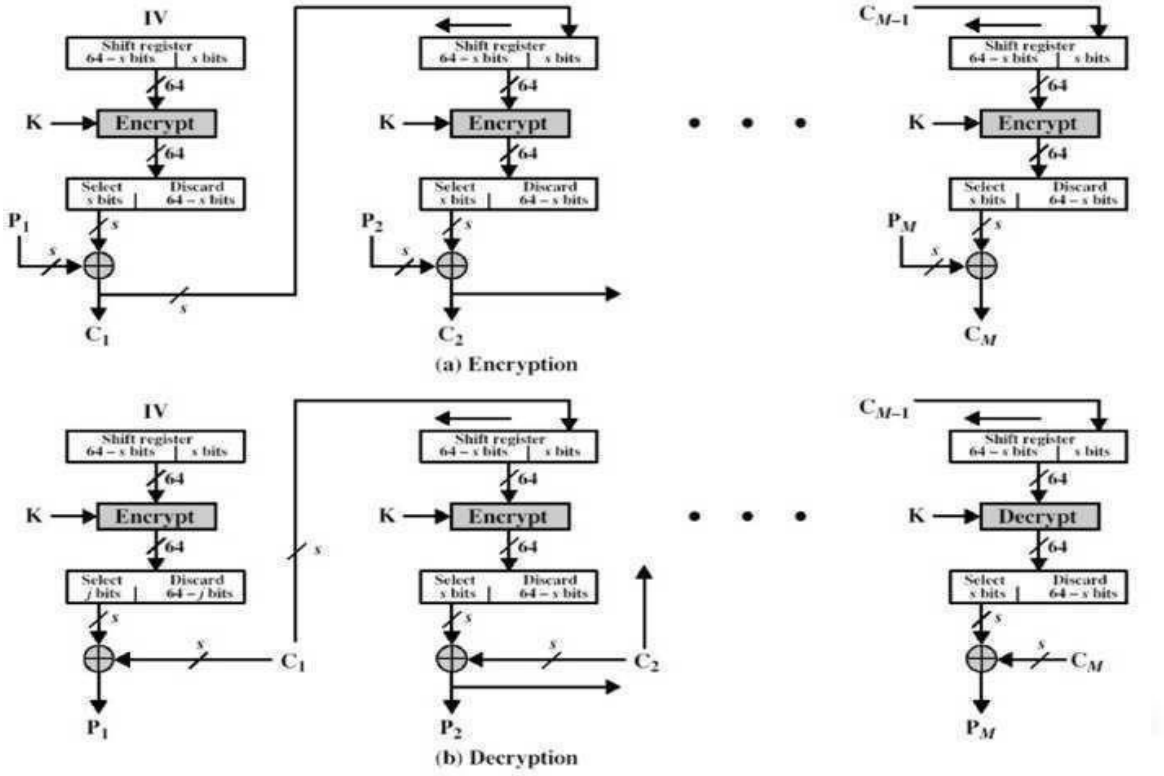
- We would like that same plaintext blocks produce different ciphertext blocks.
- Cipher Block Chaining (see figure) allows this by XORing each plaintext with the Ciphertext from the previous round (the first round using an Initialisation Vector (IV)).
- As before, the same key is used for each block.
- Decryption works as shown in the figure because of the properties ofthe XOR operation,



Cipher Feedback (CFB) Mode

- The Cipher Feedback and Output Feedback allows a block cipher to be converted into a stream cipher.
- This eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time.

• Figure shows the CFB scheme.
• In this figure it assumed that 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 (which is split into s bit segments).
• The input to the encryption function is a shift register equal in length to the block cipher of the algorithm (although the diagram shows 64 bits, which is block size used by DES, this can be extended to other block sizes such as the 128 bits of AES).
• This is initially set to some Initialisation Vector (IV).



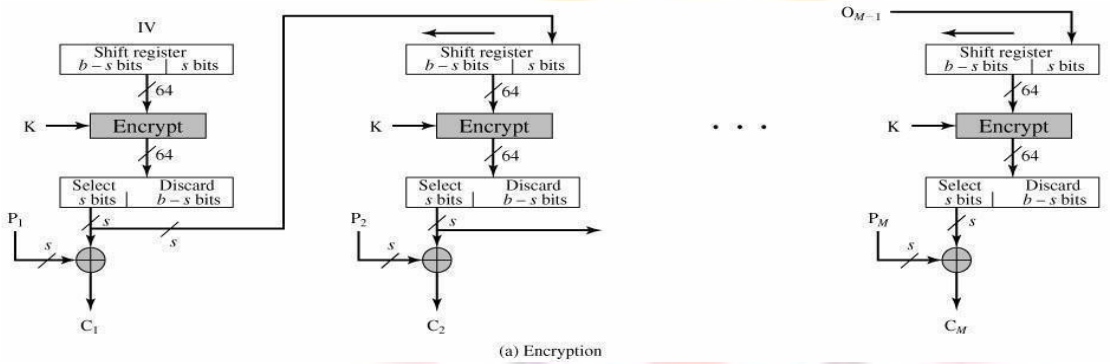
OUTPUT FEEDBACK (OFB) MODE

- The Output Feedback Mode is similar in structure to that of CFB, as seen in figure 13.
- As can be seen, it is the output of the encryption function that is fed back to the

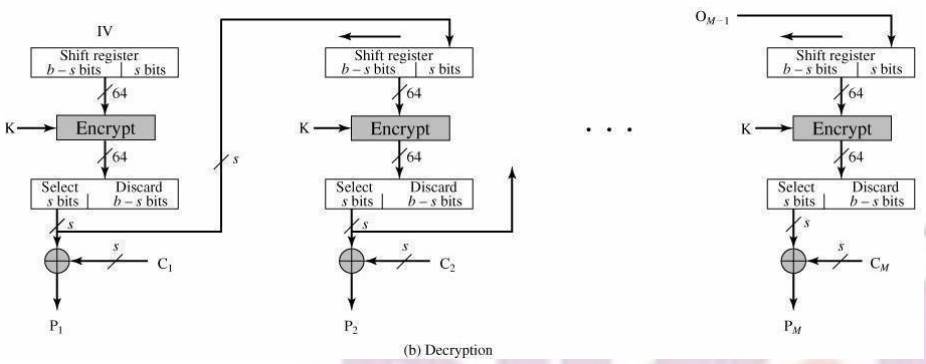
shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register.

- One advantage of the OFB method is that bit errors in transmission do not propagate.
- For example, if a bit error occurs in C1 only the recovered value of P1 is affected; subsequent plaintext units are not corrupted.

With CFB, C1 also serves as input to the shift register and therefore causes additionalcorruption downstream.

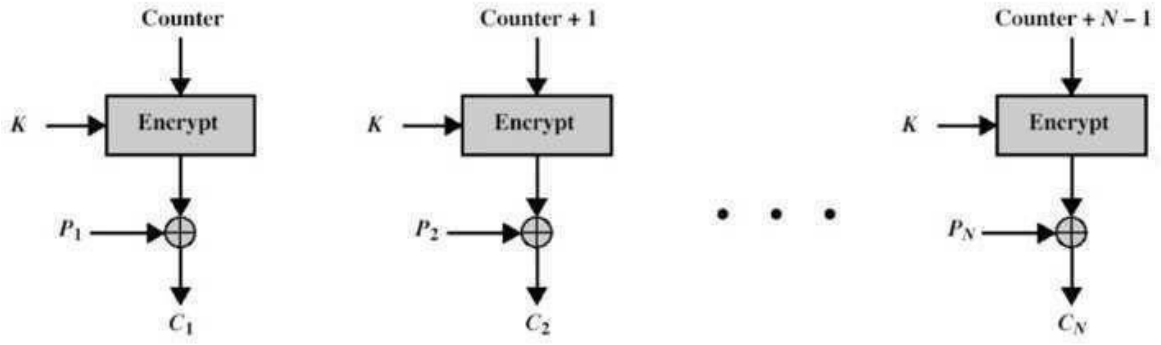


(a) Encryption

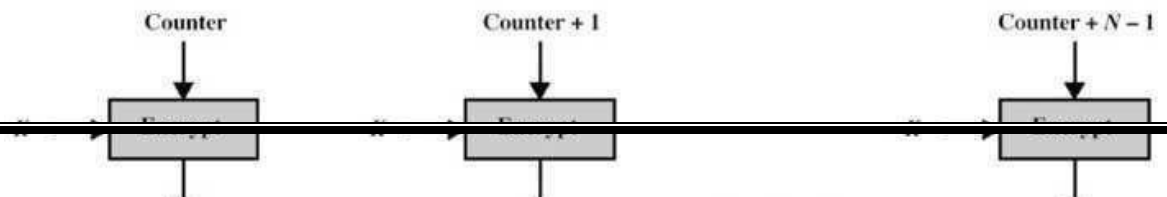


(b) Decryption

Counter Mode



(a) Encryption



INFORMATION SECURITY (CS4101PC)

8). STREAM CIPHERS

In stream cipher, one byte is encrypted at a time while in block cipher ~128 bits are encrypted at a time.

Initially, a key(k) will be supplied as input to pseudorandom bit generator and then it produces a random 8-bit output which is treated as keystream.

The resulted keystream will be of size 1 byte, i.e., 8 bits.

1. Stream Cipher follows the sequence of pseudorandom number stream.
2. One of the benefits of following stream cipher is to make cryptanalysis more difficult, so the number of bits chosen in the Keystream must be long in order to make cryptanalysis more difficult.
3. By making the key more longer it is also safe against brute force attacks.
4. The longer the key the stronger security is achieved, preventing any attack.
5. Keystream can be designed more efficiently by including more number of 1s and 0s, for making cryptanalysis more difficult.
6. Considerable benefit of a stream cipher is, it requires few lines of code compared to block cipher.

Encryption:

For Encryption,

- Plain Text and Keystream produces Cipher Text (Same keystream will be used for decryption.).
- The Plaintext will undergo XOR operation with keystream bit-by-bit and produces the Cipher Text.

Example –

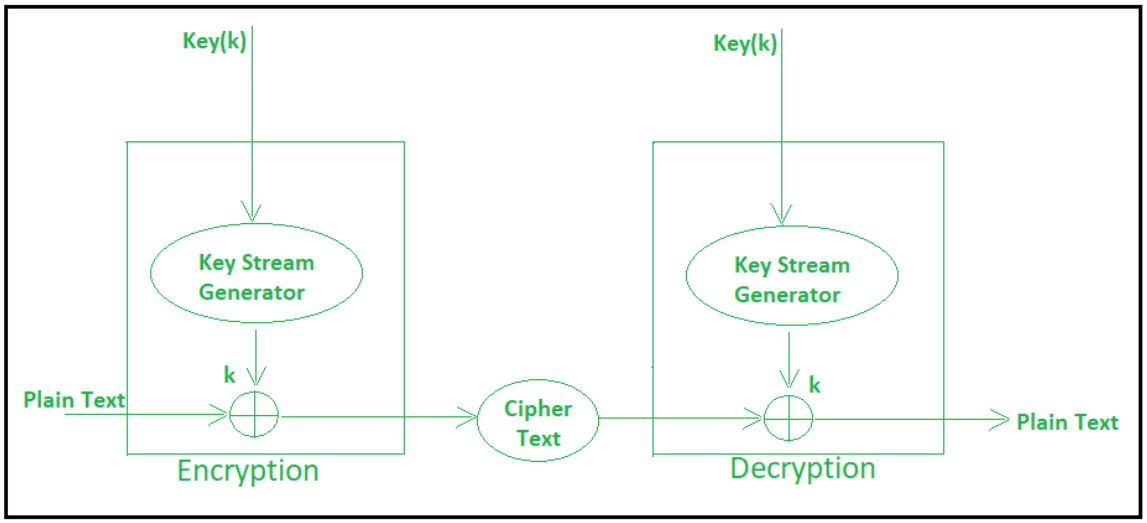
Plain Text: 10011001

Keystream: 11000011

``````````````````````

Cipher Text: 01011010

### Decryption:

For Decryption,

- Cipher Text and Keystream gives the original Plain Text (Same keystream will be used for encryption.).
- The Ciphertext will undergo XOR operation with keystream bit-by-bit and produces the actual Plain Text.

**Example –**

Cipher Text: 01011010

Keystream: 11000011

``````````````````````

Plain Text: 10011001

Decryption is just the reverse process of Encryption i.e. performing XOR with Cipher Text.



9). RC4

RC4 means Rivest Cipher 4 invented by Ron Rivest in 1987 for RSA Security. It is a Stream Ciphers. Stream Ciphers operate on a stream of data byte by byte. RC4 stream cipher is one of the most widely used stream ciphers because of its simplicity and speed of operation. It is a variable key-size stream cipher with byte-oriented operations. It uses either 64 bit or 128-bit key sizes. It is generally used in applications such as Secure Socket Layer (SSL), Transport Layer Security (TLS), and also used in IEEE 802.11 wireless LAN std.

Why Encryption Is Important?

Unauthorized data access can be prevented by encryption. If we perform encryption then third parties can not have access to data which we share or receive. The encryption is done by using a secret key, or we can say that by using a public key and private key. Both sender and receiver are having their public key and private key through which encryption of plain text and decryption of ciphertext is performed.

History of RC4 Encryption

RC4 was designed by Ron Rivest in 1987. He was working under RSA Security. Rivest Cipher 4 is an official name while it is also known as Ron's Code. Initially, RC4 was trade secret but once it's code spread in the public domain it was no more a trade secret. While Ron did not reveal the RC4 algorithm until 2014 when he described the history of RC4 in English Wikipedia.

Applications of RC4

RC4 is used in various applications such as WEP from 1997 and WPA from 2003. We also find applications of RC4 in SSL from 1995 and it is a successor of TLS from 1999. RC4 is used in varied

applications because of its simplicity, speed, and simplified implementation in both software and hardware.

Types of RC4

There are various types of RC4 such as Spritz, RC4A, VMPC, and RC4A.

1. SPRITZ: Spritz can be used to build a cryptographic hash function, a deterministic random bit generator (DRBG), n an encryption algorithm that supports authenticated encryption with associated data (AEAD).
2. RC4A: Souraduyti Paul and Bart Preneel have proposed an RC4 variant, which they call RC4A, which is stronger than RC4.
3. VMPC: VMPC is another variant of RC4 which stands for Variably Modified Permutation Composition.

RC4A+: RC4A+ is a modified version of RC4 with a more complex three-phase key schedule which takes about three times as long as RC4 and a more complex output function which performs four additional lookups in the S array for each byte output, taking approximately 1.7 times as long as basic

RC4 Algorithm

The algorithm operates on a user-selected variable-length key(K) of 1 to 256 bytes (8 to 2048 bits), typically between 5 and 16 bytes. To generate a 256-byte state vector S, the master key is used. The first step is the array initialization. It is a character array of size 256 i.e. S[256]. After that, for every element of the array, we initialize S[i] to i.

Code for array initialization:

```
Char S[256];

int i;

for(i=0;i<256;i++)

S[i] = i
```

The array will look like –

S[] = {0, 1, 2, 3, ------, 254, 255}

After this, we will run the KSA algorithm-KSA is going to use the secret key to scramble this array. KSA is a simple loop, in which we are having two variable i and j. We are using these variables to rearrange the array. Rearranging the array is done by using a secret key.

Code for KSA (Key Scheduling Algorithm) :

```
int i, j=0;
```

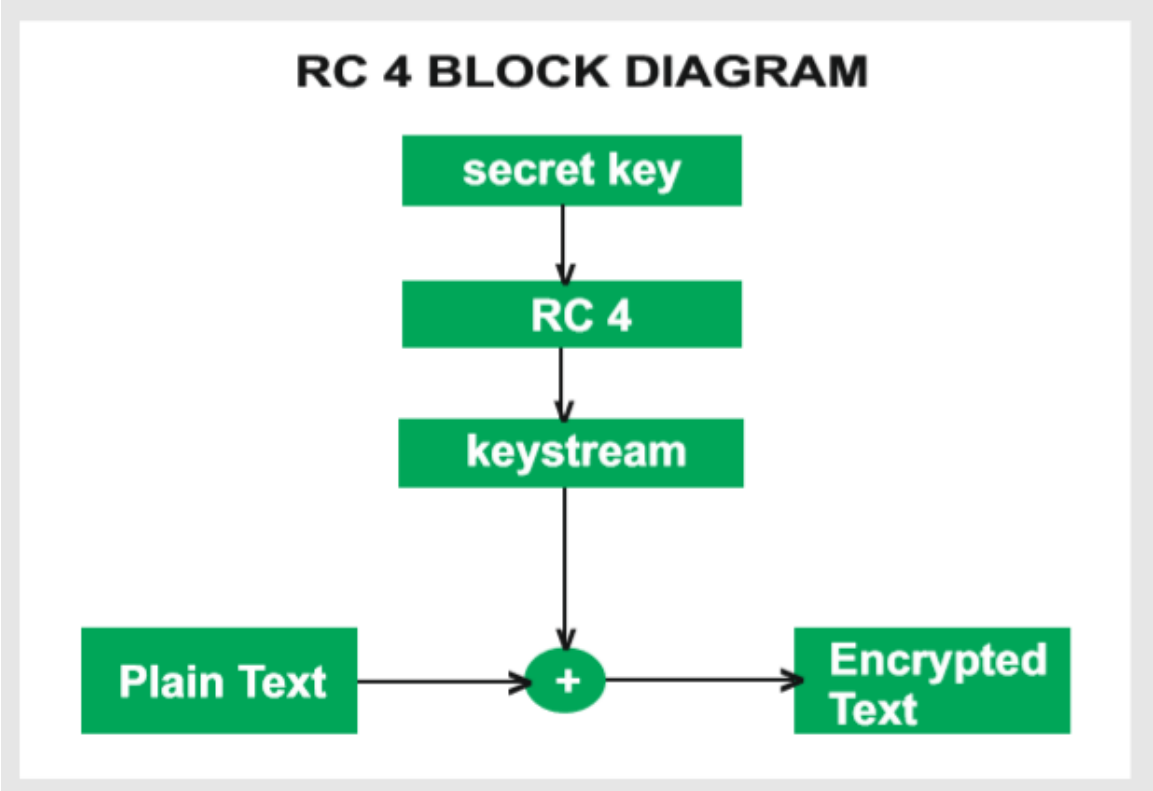
```
for(i=0;i<256;i++)
{
j=( j + S[i] + T[i]) mod 256;
Swap(S[i], S[j]);
}
```

KSA has been scrambled, S[256] array is used to generate the PRGA(Pseudo Random Generation Algorithm). This is the actual Keystream.

Code for PRGA (Pseudo Random Generation Algorithm):

```
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];
}
```

This is the next step of scrambling.



Working of RC4

Encryption Procedure

1. The user inputs a plain text file and a secret key.
2. The encryption engine then generates the keystream by using KSA and PRGA Algorithm.
3. This keystream is now XOR with the plain text, this XORing is done byte by byte to produce the encrypted text.
4. The encrypted text is then sent to the intended receiver, the intended receiver will then decrypted the text and after decryption, the receiver will get the original plain text.

Decryption Procedure

Decryption is achieved by doing the same byte-wise X-OR operation on the Ciphertext.

Example: Let A be the plain text and B be the keystream (A xor B) xor B = A

Advantages

1. RC4 stream ciphers are simple to use.
2. The speed of operation in RC4 is fast as compared to other ciphers.
3. RC4 stream ciphers are strong in coding and easy to implement.
4. RC4 stream ciphers do not require more memory.
5. RC4 stream ciphers are implemented on large streams of data.

Disadvantages

- If RC4 is not used with strong MAC then encryption is vulnerable to a bit-flipping attack.
- RC4 stream ciphers do not provide authentication.
- RC4 algorithm requires additional analysis before including new systems.
- RC4 stream ciphers cannot be implemented on small streams of data.
- RC4 fails to discard the beginning of output keystream or fails to use non-random or related keys for the algorithm.

ASSYMETRIC KEY CIPHERS

10). PRINCIPLES OF PUBLIC KEY CRYPTOSYSTEMS

The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography. It is *asymmetric*, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. Public key schemes are neither more nor less secure than private key (security depends on the key size for both). Public-key cryptography *complements rather than replaces* symmetric cryptography. Both also have issues with key distribution, requiring the useof some suitable protocol. The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

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

2.) ***digital signatures*** – how to verify a message comes intact from the claimedsender

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

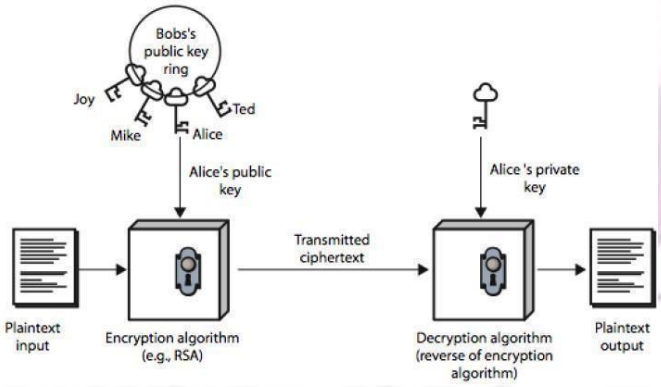
a *public-key*, which may be known by anybody, and can be used to **encrypt**

- **messages**, and **verify signatures**
- a *private-key*, known only to the recipient, used to **decrypt messages**, and **sign**
  (create) **signatures**.
- is **asymmetric** because those who encrypt messages or verifysignatures cannot decrypt messages or create signatures

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

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

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



The essential steps involved in a public-key encryption scheme are given below:

1.) Each user generates a pair of keys to be used for encryption and decryption.

2.) Each user places one of the two keys in a public register and the other key is kept private.

3.) If B wants to send a confidential message to A, B encrypts the message using A's public key.

4.) When A receives the message, she decrypts it using her private key. Nobody else can decrypt the message because that can only be done using A's private key (Deducing a private key should be infeasible).

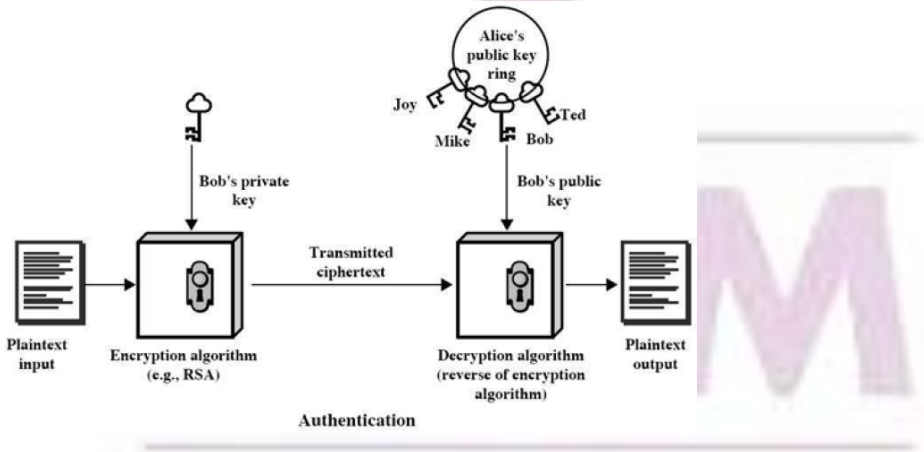
5.) If a user wishes to change his keys –generate another pair of keys and publish the public one: no interaction with other users is needed.

Notations used in Public-key cryptography:

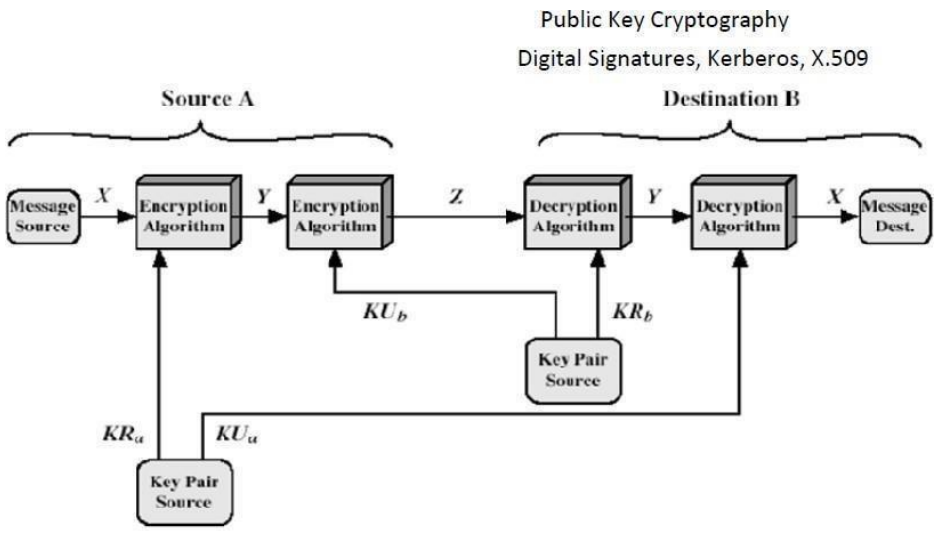
- The public key of user A will be denoted **KUA**.
- The private key of user A will be denoted **KRA**.
- Encryption method will be a function E.
- Decryption method will be a function D.
- If B wishes to send a plain message X to A, then he sends the cryptotext $Y=E(KU_A,X)$

The intended receiver A will decrypt the message: $D(KR_A,Y)=X$

The first attack on Public-key Cryptography is the attack on Authenticity. **An attacker may impersonate user B**: he sends a message $E(KU_A,X)$ and claims in the message to be B–A has no guarantee this is so. To overcome this, B will encrypt the message using his private key: $Y=E(KR_B,X)$. Receiver decrypts using B's public key KRB. This shows the authenticity of the sender because (supposedly) he is the only one who knows the private key. The entire encrypted message serves as a digital signature. This scheme is depicted in the following figure:



But, a drawback still exists. Anybody can decrypt the message using B's public key. So, secrecy or confidentiality is being compromised. One can provide both *authentication and confidentiality* using the public-key scheme twice:



Public-Key Cryptosystem: Secrecy and Authentication

B encrypts X with his private key: Y=E(KRB,X) B encrypts Y with A's public key: Z=E(KUA,Y)

A will decrypt Z (and she is the only one capable of doing it): Y=D(KRA,Z)

A can now get the plaintext and ensure that it comes from B (he is theonly one who knows his private key): decrypt Y using B's public key: X=E(KUB,Y).

Applications for public-key cryptosystems

1.) **Encryption/decryption**: sender encrypts the message with the receiver's publickey.

2.) **Digital signature**: sender "signs" the message (or a representative part of themessage) using his private key

3.) **Key exchange**: two sides cooperate to exchange a secret key for later use in asecret- key cryptosystem.

The main requirements of Public-key cryptography are:

1. Computationally easy for a party B to generate a pair (public key KUb, private key KRb).
2. Easy for sender A to generate ciphertext:
3. Easy for the receiver B to decrypt ciphertect using private key:
4. Computationally infeasible to determine private key (KRb) knowing public key (KUb)
5. Computationally infeasible to recover message M, knowing KUb and ciphertext C
6. Either of the two keys can be used for encryption, with the other usedfordecryption:

$$\mathbf{M= D_{KRb}[E_{KUb}(M)]=D_{KUb}[E_{KRb}(M)]}$$

Easy is defined to mean a problem that can be solved in polynomial time as a function of input length. A problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size. Public-key cryptosystems usually rely on difficult math functions rather than S-P networks as classical cryptosystems. **One-way function** is one, easy to calculate in one direction, infeasible to calculate in the other direction (i.e., the inverse is infeasible to compute). **Trap-door function** is a difficult function that becomes easy if some extra information is known. Our aim to find a ***trap-door one-way function***, which is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known.

Security of Public-key schemes:

- Like private key schemes brute force **exhaustive search** attack is always theoretically possible. But keys used are too large (>512bits).
- Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems. More generally the **hard** problem is known, its just made too hard to do in practise.
- Requires the use of **very large numbers,** hence is **slow** compared to private key schemes

11). RSA ALGORITHM

RSA is the best known, and by far the most widely used general public key encryption algorithm, and was first published by Rivest, Shamir & Adleman of MIT in 1978 [RIVE78]. Since that time RSA has reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption. The RSA scheme is a block cipher in which the plaintext and the ciphertext are integers between 0 and n- 1 for some fixed n and typical size for n is 1024 bits (or 309 decimal digits). It is based on exponentiation in a finite (Galois) field over integers modulo a prime, using large integers (eg. 1024 bits). Its security is due to the cost of factoring large numbers. RSA involves a public-key and a private-key where the public key is known to all and is used to encrypt data or message. The data or message which has been encrypted using a public key can only be decryted by using its corresponding private-key. Each user generates a key pair

public and private key using the following steps:

- *each user selects two large primes at random - p, q*
- *compute their system modulus n=p.q*
- *calculate ø(n), where* ***ø(n)=(p-1)(q-1)***
- *selecting at random the encryption key e, where 1<e<ø(n),and* ***gcd(e,ø(n))=1***
- *solve following equation to find decryption key d:* ***e.d=1 mod ø(n)*** *and0≤d≤n*
- *publish their public encryption key:* ***KU={e,n}***

- *keep secret private decryption key:**KR={d,n}***

Both the sender and receiver must know the values of n and e, and only the receiver knows the value of d. Encryption and Decryption are done using the following equations. To encrypt a message M the sender:

– obtains **public key** of recipient *KU={e,n}*

– computes: **C=Me mod n**, where 0≤M<n To decrypt the ciphertext C the owner:

– uses their private key *KR={d,n}*

– computes: **M=Cd mod n = (Me) d mod n = Med mod n**

For this algorithm to be satisfactory, the following requirements are to be met.

a) Its possible to find values of e, d, n such that **Med = M mod n** for all M<n

b) It is relatively easy to calculate Me and C for all values of M < n.

c) It is impossible to determine d given e andn

The way RSA works is based on Number theory: **Fermat's little theorem**: if **p** is prime and **a** is positive integer not divisible by **p**, then **ap-1 ≡ 1 mod p. Corollary**: For any positive integer **a** and prime **p**, **ap ≡ a mod p.**

Fermat's theorem, as useful as will turn out to be does not provide us with integers d,e we are looking for –Euler's theorem (a refinement of Fermat's) does. Euler's function associates to any positive integer **n,** a number **φ(n)**: the number of positive integers smaller than **n** and relatively prime to **n.** For example, **φ(37) = 36**i.e. **φ(p) = p-1** for any prime **p.** For any two primes **p,q**, **φ(pq)=(p-1)(q-1). Euler's theorem**: for any relatively prime integers a,n we have **aφ(n)≡1 mod n. Corollary: F**or any integers a,n we have **aφ(n)+1≡a mod n Corollary:** Let p,q be two odd primesand n=pq. Then: φ(n)=(p-1)(q-

1) For any integer m with 0<m<n, m(p-1)(q-1)+1 ≡ m mod n For any integers k,m with 0<m<n, mk(p-1)(q-1)+1 ≡ m mod n Euler's theorem provides us the numbers d, e such that Med=M mod n. We have to choose d,e such that ed=kφ(n)+1, or equivalently, d≡e- 1mod φ(n)

An example of RSA can be given as, Select primes: *p*=17 & *q*=11 Compute *n* = *pq*=17×11=187 Compute ø(*n*)=(*p*–1)(*q*-1)=16×10=160 Select e *:* gcd(e,160)=1; choose *e*=7 Determine d*: de*=1 mod 160 and *d* < 160 Value is d=23 since 23×7=161= 10×160+1

Publish public key KU={7,187}

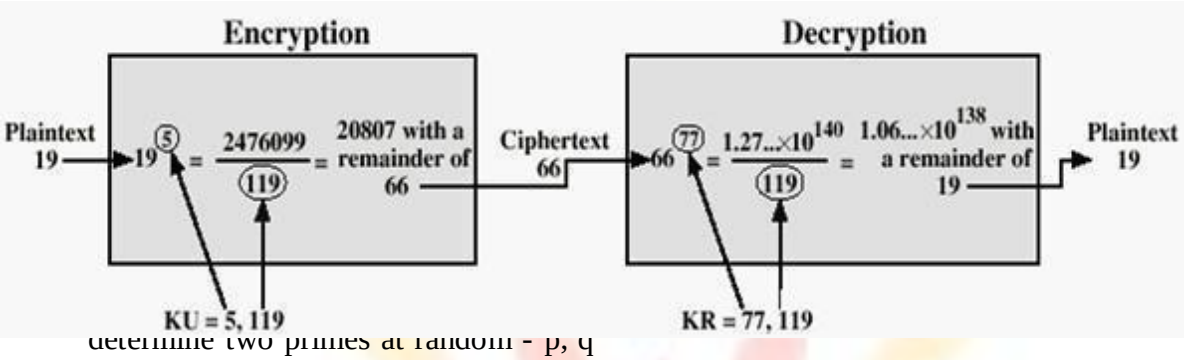
Keep secret private key KR={23,187} Now, given message M = 88 (nb. 88<187)
encryption: C = 887 mod 187 = 11
decryption: M = 1123 mod 187 = 88
Another example of RSA is given as,
Let **p = 11, q = 13, e = 11, m = 7**
n = pq i.e. n= 11*13 = 143

ø(n)= (p-1)(q-1) i.e. (11-1)(13-1) = 120

e.d=1 mod ø(n) i.e. 11d mod 120 = 1 i.e. (11*11) mod 120=1; so d = 11 public key :{11,143} and private key: {11,143}

C=Me mod n, so ciphertext = 711mod143 = 727833 mod 143; i.e. C = 106

M=Cd mod n, plaintext = 10611 mod 143 = 1008 mod 143; i.e. M = 7



determine two primes at random - p, q

- select either e or d and compute the other

- means must be sufficiently large
- typically guess and use probabilistic test

Security of RSA

There are three main approaches of attacking RSA algorithm.

Brute force key search (infeasible given size of numbers) As explained before, involves trying all possible private keys. Best defence is using large keys.

Mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N) There are several approaches, all equivalent in effect to factoring the product of two primes. Some of them are given as:

- factor N=p.q, hence find ø(N) and then d
- determine ø(N) directly and find d
- find d directly

The possible defense would be using large keys and also choosing large numbers for p and q, which should differ only by a few bits and are also on the order of magnitude 10_{75} to 10_{100}. And gcd (p-1, q-1) should be small.

12). ELGAMAL ENCRYPTION ALGORITHM

Elgamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message. This cryptosystem is based on the difficulty of finding **discrete logarithm** in a cyclic group that is even if we know g^a and g^k, it is extremely difficult to compute g^{ak}.

Idea of Elgamal cryptosystem

Suppose Alice wants to communicate with Bob.

1. Bob generates public and private keys:
   - Bob chooses a very large number **q** and a cyclic group $\mathbf{F}_q$.
   - From the cyclic group $\mathbf{F}_q$, he choose any element **g** and an element **a** such that gcd(a, q) = 1.
   - Then he computes $h = g^a$.
   - Bob publishes **F**, $\mathbf{h = g^a}$, **q**, and **g** as his public key and retains **a** as private key.
2. Alice encrypts data using Bob's public key :
   - Alice selects an element **k** from cyclic group **F** such that gcd(k, q) = 1.
   - Then she computes $p = g^k$ and $s = h^k = g^{ak}$.
   - She multiples s with M.
   - Then she sends $(p, M*s) = (g^k, M*s)$.
3. Bob decrypts the message :
   - Bob calculates $s' = p^a = g^{ak}$.
   - He divides M*s by s' to obtain M as $s = s'$.

Following is the implementation of the ElGamal cryptosystem in Python Python3

```
# Python program to illustrate Elgamal encryption

import random
from math import pow

a = random.randint(2, 10)

def gcd(a, b):
    if a < b:
```

```
        return gcd(b, a)
    elif a % b == 0:
        return b;
    else:
        return gcd(b, a % b)

# Generating large random numbers
def gen_key(q):

    key = random.randint(pow(10, 20), q)
    while gcd(q, key) != 1:
        key = random.randint(pow(10, 20), q)

    return key

# Modular exponentiation
def power(a, b, c):
    x = 1
    y = a

    while b > 0:
        if b % 2 != 0:
            x = (x * y) % c;
        y = (y * y) % c
        b = int(b / 2)

    return x % c
```

```
# Asymmetric encryption
def encrypt(msg, q, h, g):

    en_msg = []

    k = gen_key(q)# Private key for sender
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
        en_msg.append(msg[i])

    print("g^k used : ", p)
    print("g^ak used : ", s)
    for i in range(0, len(en_msg)):
        en_msg[i] = s * ord(en_msg[i])

    return en_msg, p

def decrypt(en_msg, p, key, q):

    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
        dr_msg.append(chr(int(en_msg[i]/h)))
```

```
    return dr_msg


# Driver code
def main():

    msg = 'encryption'
    print("Original Message :", msg)


    q = random.randint(pow(10, 20), pow(10, 50))
    g = random.randint(2, q)

    key = gen_key(q)# Private key for receiver
    h = power(g, key, q)
    print("g used : ", g)
    print("g^a used : ", h)

    en_msg, p = encrypt(msg, q, h, g)
    dr_msg = decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr_msg)
    print("Decrypted Message :", dmsg);


if __name__ == '__main__':
    main()
```

SampleOutput :

Original Message : encryption

g used : 586069695452241770718895237154794403533331590789O

g^a used : 47113097556393642895524548345062151446539580552 52

g^k used : 12475188089503227615789015740709091911412567126782

g^ak used : 39448787632167136161153337226654906357756740068295

Decrypted Message : encryption

In this cryptosystem, the original message **M** is masked by multiplying $\mathbf{g^{ak}}$ to it. To remove the mask, a clue is given in form of $\mathbf{g^k}$. Unless someone knows **a**, he will not be able to retrieve **M**. This is because finding discrete log in a cyclic group is difficult and simplifying knowing g^a and g^k is not good enough to compute g^{ak}.

13). DIFFIE-HELLMAN KEY EXCHANGE

Diffie-Hellman key exchange (**D-H**) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. The D-H algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

First, a primitive root of a prime number p, can be defined as one whose powers generate all the integers from 1 to p-1. If a is a primitive root of the prime number p, then the numbers, ***a mod p, a_2 mod p,..., a_{p-1} mod p***, are distinct and consist of the integers from 1 through p 1 in some permutation.

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

i such that $b \equiv a^i \pmod{p}$ where $0 \leq i \leq (p\ 1)$ .The exponent i is referred to as thediscrete logarithm of b for the base a, mod p. We express this value as ***$dlog_{a,p}$ (b).*** The algorithm is summarized below:

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

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A
$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B
$K = (Y_A)^{X_B} \bmod q$

For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha XA \bmod q$. Similarly, user B independently selects a random integer $X_A < q$ and computes $Y_B = \alpha XB \bmod q$.
Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)_{XA} \bmod q$ and user B computes the key as $K = (Y_A)_{XB} \bmod$

Discrete Log Problem

The (discrete) exponentiation problem is as follows: Given a base a, an exponent b and a modulus p, calculate c such that $a_b \equiv c \pmod p$ and $0 \leq c < p$. It turns out that this problem is fairly easy and can be calculated "quickly" using fast-exponentiation. The discrete log problem is the inverse problem: Given a base a, a result c ($0 \leq c < p$) and a modulus p,calculate the exponent b such that ab ≡ c (mod p). It turns out that no one has found a quick way to solve this problem With DLP, if P had 300 digits, X_a and X_b have more than 100 digits, it would take longer than the life of the universe to crack the method.

Examples for D-H key distribution scheme:

1) Let p = 37 and g = 13.

Let Alice pick a = 10. Alice calculates 13_{10} (mod 37) which is 4 and sends that to Bob. Let Bob pick b = 7. Bob calculates 13_7 (mod 37) which is 32 and sends that to Alice. (Note: 6 and 7 are secret to Alice and Bob, respectively, but both 4 and 32 are known by all.)

10 (mod 37) which is 30, the secret key.
7 (mod 37) which is 30, the same secret key.

2) Let p = 47 and g = 5. Let Alice pick a = 18. Alice calculates 5_{18} (mod 47) which is 2 and sends that to Bob. Let Bob pick b = 22. Bob calculates 5_{22} (mod 47) which is 28 and sends that to Alice.

18 (mod 47) which is 24, the secret key.
22 (mod 47) which is 24, the same secret key

Man-in-the-Middle Attack on D-H protocol

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} andY_{D2}.
2. Alice transmits Y_A toBob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates K2 =

$(Y_A)_{XD2}$modq.

4. Bob receives Y_{D1} and calculates K1 = $(Y_{D1})_{XE}$ mod q.
5. Bob transmits X_A toAlice.
6. Darth intercepts X_A and transmits Y_{D2} to Alice. Darth calculates K1 = $(Y_B)_{XD1}$ mod q.
7. Alice receives Y_{D2} and calculates K2 = $(Y_{D2})_{XA}$ modq.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message M: E(K2, M).
2. Darth intercepts the encrypted message and decrypts it, to recover M.
3. Darth sends Bob E(K1, M) or E(K1, 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.

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

14). KNAPSACK ENCRYPTION ALGORITHM

Knapsack Encryption Algorithm is the first general public key cryptography algorithm. It is developed by **Ralph Merkle** and **Mertin Hellman** in 1978. As it is a Public key cryptography, it needs two different keys. One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process. In this algorithm we will two different knapsack problems in which one is easy and other one is hard. The easy knapsack is used as the private key and the hard knapsack is used as the public key. The easy knapsack is used to derived the hard knapsack.

For the easy knapsack, we will choose a **Super Increasing knapsack problem**. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

Example –

{1, 2, 4, 10, 20, 40} is a super increasing as

1<2, 1+2<4, 1+2+4<10, 1+2+4+10<20 and 1+2+4+10+20<40.

1.Encryption :

As our knapsacks contain six values, so we will split our plain text in a groups of six:

100100 111100 101110

Multiply each values of public key with the corresponding values of each group and take their sum.

100100 {31, 62, 14, 90, 70, 30}

1x31+0x62+0x14+1x90+0x70+0x30 = 121

111100 {31, 62, 14, 90, 70, 30}

1x31+1x62+1x14+1x90+0x70+0x30 = 197

101110 {31, 62, 14, 90, 70, 30}

1x31+0x62+1x14+1x90+1x70+0x30 = 205

So, our cipher text is 121 197 205.

2.Decryption :

The receiver receive the cipher text which has to be decrypt. The receiver also knows the values of m and n.

So, first we need to find the , which is multiplicative inverse of n mod m i.e.,

n x mod(m) = 1

31 x mod(110) = 1

Now, we have to multiply 71 with each block of cipher text take modulo m.

121 x 71 mod(110) = 11

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e., 1+10=11 so make that corresponding bits 1 and others 0 which is 100100. Similarly,

197 x 71 mod(110) = 17

1+2+4+10=17 = 111100

And, 205 x 71 mod(110) = 35

1+4+10+20=35 = 101110

After combining them we get the decoded text.

100100111100101110 which is our plai

Descriptive Questions:

(a) 2 Marks Questions

- **Define cryptanalysis and cryptology.**

Cryptanalysis: techniques used for deciphering or decrypting a message without the knowledge of the enciphering or encrypting details is said to be cryptanalysis. Cryptology: the study of cryptography and cryptanalysis together is called cryptology.

2. Mention the various types of cryptanalytic attack.

- Known plaintext
- Cipher text only
- Chosen plaintext
- Chosen Cipher text

3. Define symmetric key cryptography and public key cryptography.

In symmetric key cryptography, only one key is used for encryption and decryption.

In public key cryptography, two keys (public key and private key) are used. When one key is used for encryption, then the other must be used for decryption. The public key is known to all the participants but the private key is kept secret by the owner.

4. Define Primality Test.

A primality testing is a test to determine whether or not a given number is prime, as opposed to actually decomposing the number into its constituent prime factors (which is known as prime factorization).

5. Define Euler's totient function (used in RSA algorithm).

It is the number of positive integers that are less than 'n' and relatively prime to 'n'. Where n is the product of two prime numbers (p & q)

It is represented as $\Phi(n)$ and it is expressed as $\Phi(n) = \Phi(pq) = (p-1)(q-1)$.

6. What are the various approaches to attacks the RSA algorithm?

- Brute force attack
- Mathematical attacks
- Timing attacks

7. Define Euler's theorem and it's application.

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

8. Find gcd (1970, 1066) using Euclid's algorithm?

gcd (1970,1066) = gcd(1066,1970 mod 1066)
= gcd(1066,904)= 2

9. Find gcd (24140, 16762) using Euclid's algorithm?

gcd (24140,16762) = gcd(16762, 24140 mod 16762)
= gcd(16762 , 7378)= 34

Why do we need Diffie Hellman algorithm?

It is used for exchanging the secret keys between the sender and the receiver. It allows twousers to exchange a key securely.

11. What is an elliptic curve?

It is a plane algebraic curve defined by an equation of the form $y2 = x3 + ax + b$ that is non-singularalso graph has no cusps or self intersections

(b) 10 Marks Questions

1. Write short notes on:i)Linear congruence ii) Exponential and discrete logarithm

2. A) Explain in detail about Elgamal Cryptosystem and Chinese Remainder theorem
   B) In RSA system, the public key of given user e=31,n=3599 what is the private key of user?

3. A) What are the different algorithms in which prime numbers are used?
   B) Explain RSA algorithm. And perform Encryption and Decryption using RSAp=3q=11 e=7 M=5

4. Explain Diffie-Hellman key exchange algorithm in detail.

5. A) What are the requirements of public key cryptography?
   B) Explain the symmetric key cryptography and asymmetric key cryptography with an example.

6. Find the result of $(x^5 + x^2 + x)y(x^7 + x^4 + x^3 + x^2 + x)$in GF$(2^8)$ with irreducible polynomial $x^8 + x^4 + x^3 + x + 1$

7. Explain process and Algorithm of Extended Euclidean for GCD? Given a=161 b=28,findgcd(a,b) and the values of s and t by using Extended Euclidean Algorithm?

8. A) Explain Euler's phi Function and Fermat's little theorem?
   B) Explain Elliptic curve cryptography and Miller Rabin algorithm with an example proble m

6.

UNIT-3

Cryptographic Hash Functions: Message Authentication, Secure Hash Algorithm (SHA-512), **Message authentication codes:** Authentication requirements, HMAC, CMAC, Digital signatures, Elgamal Digital Signature Scheme.

Key Management and Distribution: Symmetric Key Distribution Using Symmetric & Asymmetric Encryption, Distribution of Public Keys, Kerberos, X.509 Authentication Service, Public–Key Infrastructure

CRYPTOGRAPHIC HASH FUNCTIONS

1). MESSAGE AUTHENTICATION

Data is prone to various attacks. One of these attacks includes message authentication. This threat arises when the user does not have any information about the originator of the message. Message authentication can be achieved using cryptographic methods which further make use of keys.

- **Revelation:** It means releasing the content of the message to someone who does not have an appropriate cryptographic key.
- **Analysis of Traffic:** Determination of the pattern of traffic through the duration of connection and frequency of connections between different parties.
- **Deception:** Adding out of context messages from a fraudulent source into a communication network. This will lead to mistrust between the parties communicating and may also cause loss of critical data.
- **Modification in the Content:** Changing the content of a message. This includes inserting new information or deleting/changing the existing one.
- **Modification in the sequence:** Changing the order of messages between parties. This includes insertion, deletion, and reordering of messages.
- **Modification in the Timings:** This includes replay and delay of messages sent between different parties. This way session tracking is also disrupted.
- **Source Refusal:** When the source denies being the originator of a message.
- **Destination refusal:** When the receiver of the message denies the reception.

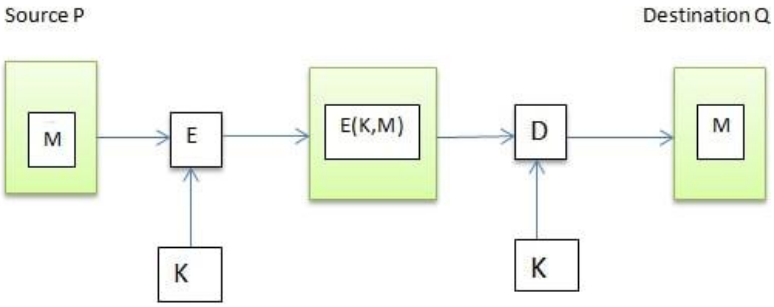
Message Authentication Functions:

All message authentication and digital signature mechanisms are based on two functionality levels:

- **Lower level:** At this level, there is a need for a function that produces an authenticator, which is the value that will further help in the authentication of a message.
- **Higher-level:** The lower level function is used here in order to help receivers verify the authenticity of messages.

These message authentication functions are divided into three classes:

- **Message encryption:** While sending data over the internet, there is always a risk of a Man in the middle(MITM) attack. A possible solution for this is to use message encryption. In message encryption, the data is first converted to a ciphertext and then sent any further. Message encryption can be done in two ways:
- **Symmetric Encryption:** Say we have to send the message M from a source P to destination Q. This message M can be encrypted using a secret key K that both P and Q share. Without this key K, no other person can get the plain text from the ciphertext. This maintains confidentiality. Further, Q can be sure that P has sent the message. This is because other than Q, P is the only party who possesses the key K and thus the ciphertext can be decrypted only by Q and no one else. This maintains authenticity. At a very basic level, symmetric encryption looks like this:

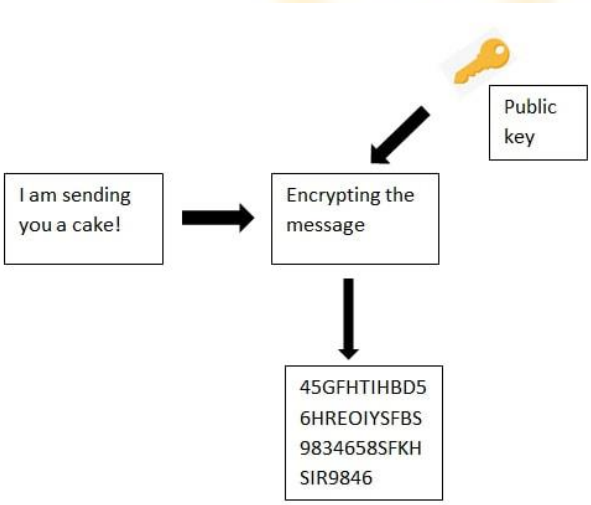


- **Public key Encryption:** Public key encryption is not as advanced as symmetric encryption as it provides confidentiality but not authentication. To provide both authentication and confidentiality, the private key is used.
- **Message authentication code (MAC):** A message authentication code is a security code that the user of a computer has to type in order to access any account or portal. These codes are recognized by the system so that it can grant access to the right user. These codes help in maintaining information integrity. It also confirms the authenticity of the message.
- **Hash function:** A hash function is nothing but a mathematical function that can convert a numeric value into another numeric value that is compressed. The input to this hash function can be of any length but the output is always of fixed length. The values that a hash function returns are called the message digest or hash values.

Measures to deal with these attacks:

Each of the above attacks has to be dealt with differently.

Message Confidentiality: To prevent the messages from being revealed, care must be taken during the transmission of messages. For this, the message should be encrypted before it is sent over the network.



- **Message Authentication:** To deal with the analysis of traffic and deception issues, message authentication is helpful. Here, the receiver can be sure of the real sender and his identity. To do this, these methods can be incorporated:
    - Parties should share secret codes that can be used at the time of identity authentication.
    - Digital signatures are helpful in the authentication.
    - A third party can be relied upon for verifying the authenticity of parties.
- **Digital Signatures:** Digital signatures provide help against a majority of these issues. With the help of digital signatures, content, sequence, and timing of the messages can be easily monitored. Moreover, it also prevents denial of message transmission by the source.
- **Combination of protocols with Digital Signatures:** This is needed to deal with the denial of messages received. Here, the use of digital signature is not sufficient and it additionally needs protocols to support its monitoring.

MESSAGE AUTHENTICATION CODES

Message authentication is a procedure to verify that received messages come from the alleged

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

Message encryption–the ciphertext serves as authenticator

Message authentication code (MAC)–a public function of the message and a secret key producing a fixed-length value to serve as authenticator. This does not provide a digital signature because A and B share the same key.

Hash function–a public function mapping an arbitrary length message into a

fixed- length hash value to serve as authenticator. This does not provide a digital signature because there is no key.

MESSAGE ENCRYPTION:

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

Sometimes one needs to avoid encryption of full messages due to legal requirements.

Encryption and authentication may be separated in the system architecture.

The different ways in which message encryption can provide authentication, confidentiality in both symmetric and asymmetric encryption techniques is explained with the table below:

Confidentiality and Authentication Implications of Message Encryption

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

MESSAGE AUTHENTICATION CODE

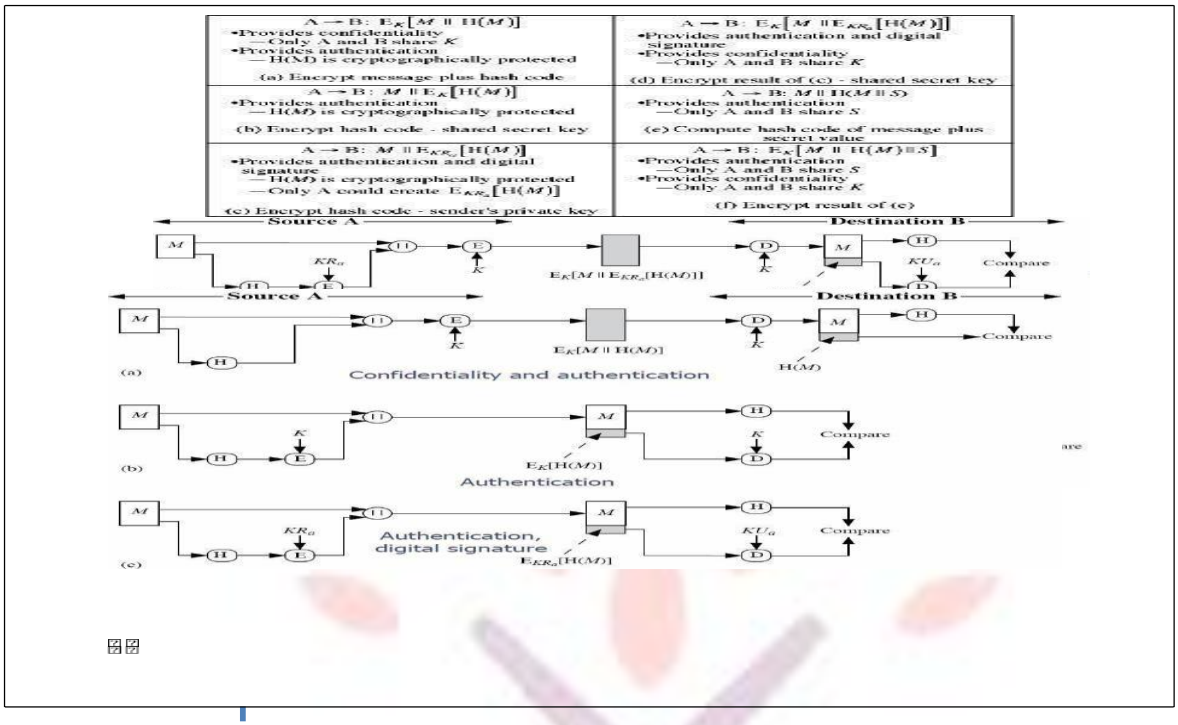
An alternative authentication technique involves the use of a secrete key to generate a small fixed- size block of data, known as cryptographic checksum or MAC, which is appended to the message. This technique assumes that both the communicating parties say A and B share a common secret key K. When A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

MAC=Ck(M) The message and the MAC are transmitted to the intended recipient, who upon receiving performs the same calculation on the received message, using the same secret key to generate a new MAC. The received MAC is compared to the calculated MAC and only if they match, then:

1. The receiver is assured that the message has not been altered: Any alternations been done the MAC's do not match.
2. The receiver is assured that the message is from the alleged sender: No oneexcept the sender has the secret key and could prepare a message with a proper MAC.
3. If the message includes a sequence number, then receiver is assured of proper

sequence as an attacker cannot successfully alter the sequence number.

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



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

If a message is broadcast to several destinations in a network (such as a military control center), then it is cheaper and more reliable to have just one node responsible to evaluate the authenticity –message will be sent in plain with an attached authenticator.

If one side has a heavy load, it cannot afford to decrypt all messages –it will just check the authenticity of some randomly selected messages.

Authentication of computer programs in plaintext is very attractive service as they need not be decrypted every time wasting of processor resources. Integrity of the program can always be checked by MAC.

MESSAGE AUTHENTICATION CODE BASED ON DES

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

The data (e.g., message, record, file, or program) to be authenticated are grouped into

The data (e.g., message, record, file, or program) to be authenticated are grouped into

Contiguous 64-bit blocks: D1, D2,..., DN. If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm, E, and a secret key, K, a data authentication code (DAC) is calculated asfollows:
The DAC consists of either the entire block ON or the leftmost M bits of the block, with $16 \leq M \leq 64$ Use of MAC needs a shared secret key between the communicatingparties and also MAC does not provide digital signature. The following table summarizes the confidentiality and authentication implications of theapproachesshown above.

HASH FUNCTION

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

In case where confidentiality is not required methods b and c have an advantage over those that encrypt the entire message in less that computation is required. Growing interest for technique that avoid encryption is due to reason like encryption software is quiet slow and may be covered by patents. Also encryption hardware costs are not negligible and the algorithms are subjected to U.S export control. A fixed-length hash value h is generated by a function H that takes as input a message of arbitrary length**: h=H(M).**sends M and H(M)

A authenticates the message by computing H(M) and checking the match B

Requirements for a hash function: The purpose of a hash function is to produce a"fingerprint" of a file, message, or other block of data. To be used for message authentication, the hash function H must have the following properties can be applied to a message of any size produces fixed-length output

H Computationally easy to compute H(M) for any given M
Computationally infeasible to find M such that H(M)=h, for a given h, referred to as
The *one-way property*
Computationally infeasible to find M' such that H(M')=H(M), for a given M, referredto as
weak collision resistance.

Computationally infeasible to find M,M' with H(M)=H(M') (to resist to birthday attacks), referred to as *strong collision resistance*.

Examples of simple hash functions are:

- Bit-by-bit XOR of plaintext blocks: h= D1⊕D2⊕…⊕DN
- Rotated XOR –before each addition the hash value is rotated to the left with 1 bit
- Cipher block chaining technique without a secret key.

2). SECURE HASH ALGORITHM

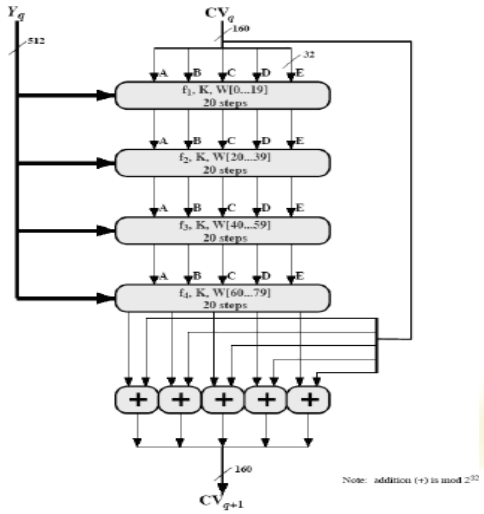
The secure hash algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST). SHA-1 is the best established of the existing SHA hash functions, and is employed in several widely used security applications and protocols. The algorithm takes as input a message with a maximum length of less than 264 bits and produces as output a 160-bit message digest The input is processed in 512-bit blocks. The overall processing of a message follows the structure of MD5 with block length of 512 bits and a hash length and chaining variable length of 160 bits. The processing consists of following steps:

1.) ***Append Padding Bits:*** The message is padded so that length is congruent to 448 modulo 512; padding always added –one bit 1 followed by the necessary number of 0 bits.

2.) ***Append Length:*** a block of 64 bits containing the length of the original message is added. 3.) ***Initialize MD buffer*****:** A 160-bit buffer is used to hold intermediate and final results on the hash function. This is formed by 32-bit registers A,B,C,D,E. Initial values: A=0x67452301, B=0xEFCDAB89, C=0x98BADCFE, D=0x10325476, E=C3D2E1F0. Stores
in big-endian format
i.e. the most significant bit in low address.

4.) ***Process message in blocks 512-bit (16-word) blocks*****:** The processing of a single 512-bit block is shown above. It consists of four rounds of processing of 20 steps each. These four rounds have similar structure, but uses a different primitive logical function, which we refer to as f1, f2, f3 and f4. Each round takes as input the current 512-bit block being processed and the 160-bit buffer value ABCDE and updates the contents of the buffer. Each round also makes use of four distinct additive constants Kt. The output of the fourth round i.e. eightieth step is added to the input to the first round to produce CV_{q+1}.

5.) *Output:* After all L 512-bit blocks have been processed, the output from the Lth stage is the 160-bit message digest.

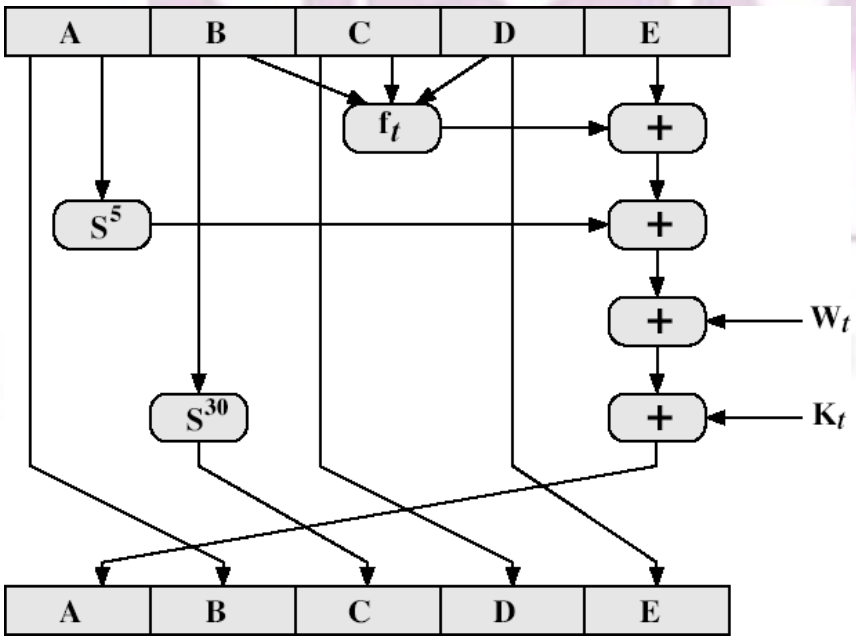


SHA-1 Processing of a Single 512-bit Block
(SHA-1 Compression Function)

The behavior of SHA-1 is as follows: **CV0 = IV CVq+1 = SUM32(CVq, ABCDEq) MD = CVL** Where, IV = initial value of ABCDE buffer ABCDEq = output of last round of processing of qth message block L = number of blocks in the message SUM32 = Addition modulo 232 MD = final message digest value.

SHA-1 Compression Function:

Each round has 20 steps which replaces the 5 buffer words. The logic present in each one of the 80 rounds present is given as **(A,B,C,D,E) <- (E + f(t,B,C,D) + S_5(A)+ Wt+ Kt),A,S_{30}(B),C,D** Where, A, B, C, D, E = the five words of the buffer t = step number; 0<t< 79 f(t,B,C,D) = primitive logical function for step t S_k = circular left shift of the 32-bit argument by k bits Wt = a 32-bit word derived from current 512-bit input block. Kt = an additive constant; four distinct values are used + = modulo addition.



SHA shares much in common with MD4/5, but with 20 instead of 16 steps in each of the 4 rounds. Note the 4 constants are based on sqrt(2,3,5,10). Note also that instead of just splitting the input block into 32-bit words and using them directly, SHA-1 shuffles and mixes them using rotates & XOR's to form a more complex input, and greatly increases the difficulty of finding collisions. A sequence of logical functions f0, f1,..., f79 is used in the SHA-1. Each f**t**, 0<=t<=79, operates on three 32-bit words B, C, D and produces a 32-bit word as output. ft(B,C,D) is defined as follows: for words B, C, D, **ft(B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19) ft(B,C,D) = B XOR C XOR D (20 <= t <= 39) ft(B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <=59) ft(B,C,D) = B XOR C XOR D (60 <= t <= 79).**

3). AUTHENTICATION REQUIREMENTS

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

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

Measures to deal with the first two attacks are in the realm of message confidentiality and are dealt with in Part One. Measures to deal with items 3 through 6 in the foregoing list are generally regarded as message authentication. Mechanisms for dealing specifically with item 7 come under the heading of

digital signatures. Generally, a digital signature technique will also counter some or all of the attacks listed under items 3 through 6. Dealing with item 8 may require a combination of the use of digital signatures and a protocol designed to counter this attack.

In summary, message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

4). HMAC

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

HMAC Design Objectives

To use, without modifications, available hash functions To allow for easy replaceability of the embedded hash function To preserve the original performance of the hash function To use and handle keys in a simple wayTo have a well understood cryptographic analysis of the strength of the MAC based on reasonable assumptions on the embedded hash function The first two objectives are very important for the acceptability of HMAC. HMAC treats the hash function as a "black box", which has two benefits. First is that an existing implementation of the hash function can be used for implementing HMAC making the bulk of HMAC code readily available without modification. Second is that if ever an existing hash function is to be replaced, the existing hash function module is removed and new module is dropped in. The last design objective provides the main advantage of HMAC over other proposed hash-based schemes. HMAC can be proven secure provided that the embedded hash function has some reasonable cryptographic strengths.

Steps involved in HMAC algorithm:

1. Append zeroes to the left end of K to create a b-bit string K_+ (ex: If K is of length 160- bits and b = 512, then K will be appended with 44zero bytes).
2. XOR(bitwise exclusive-OR) K_+ with ipad to produce the b-bit block S_i.
3. Append M to S_i.
4. Now apply H to the stream generated in step-3
5. XOR K_+ with opad to produce the b-bit block S_0.
6. Append the hash result from step-4 to S_0.

7. Apply H to the stream generated in step-6 and output the result.

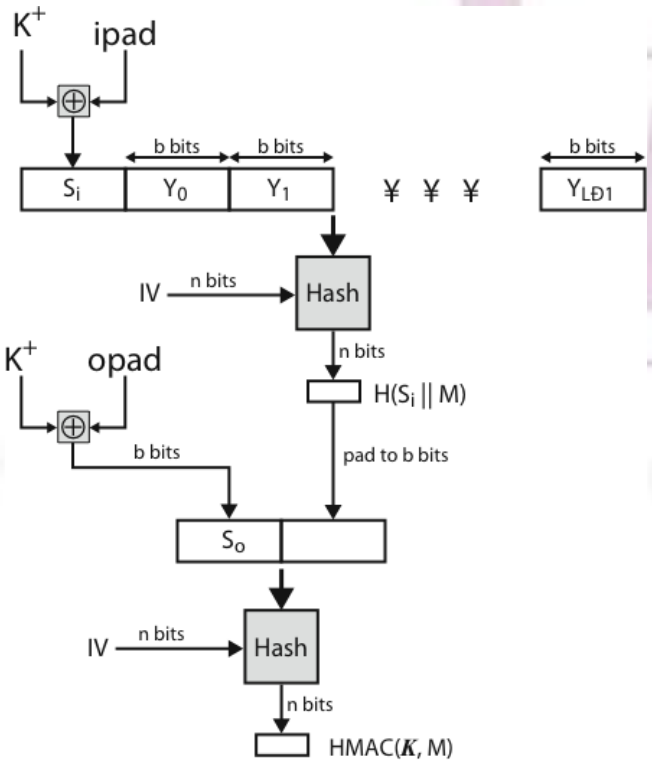
HMAC Algorithm

- **Define the following terms**

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

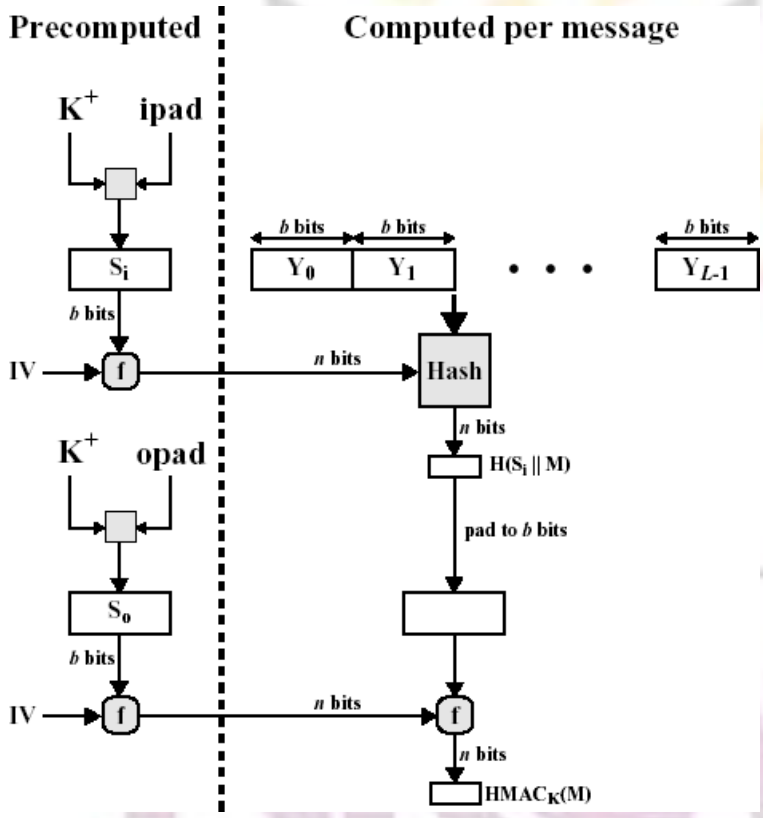
- **Then HMAC can be expressed as**

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



The XOR with ipad results in flipping one-half of the bits of K. Similarly, XOR with opad results in flipping one-half of the bits of K, but different set of bits. By passing S_i and S_0 through the compression function of the hash algorithm, we have pseudo randomly generated two keys from K. HMAC should execute in approximately the same time as the embedded hash function for long messages. HMAC adds three executions of the hash compression function (for S_0, S_i, and the block produced from the inner hash)

A more efficient implementation is possible. Two quantities are precomputed. f(IV, (K+f(IV, (K+where f is the compression function for the hash function which takes as arguments a chaining variable of n bits and a block of b-bits and produces a chaining variable of n bits.



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

The appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC. The security of a MAC function is generally expressed in terms of the probability of

successful forgery with a given amount of time spent by the forger and a given number of message- MAC pairs created with the same key. Have two classes of attacks on the embedded hash function:

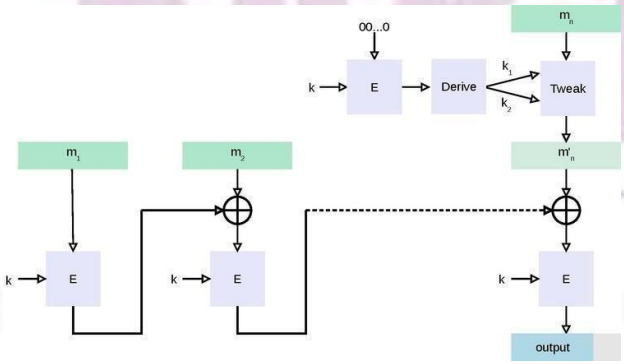
1. The attacker is able to compute an output of the compression function
2. even with anIV that is random, secret and unknown to the attacker.
3. The attacker finds collisions in the hash function even when the IV is random and secret.

These attacks are likely to be caused by brute force attack on key used which has work of order 2_n; or a birthday attack which requires work of order $2_{(n/2)}$ - but which requires the attacker to observe 2_n blocks of messages using the same key - very unlikely. So even MD5 is still secure for use in HMAC given these constraints.

5). CMAC

In cryptography, **CMAC** (Cipher-based Message Authentication Code)[1] is a block cipher- based message authentication code algorithm. It may be used to provide assurance of the authenticity and, hence, the integrity of binary data. This mode of operation fixes security deficiencies of CBC-MAC (CBC-MAC is secure only for fixed- length messages).

The core of the CMAC algorithm is a variation of CBC-MAC that Black and Rogaway proposed and analyzed under the name XCBC[2] and submitted to NIST.[3] The XCBC algorithm efficiently addresses the security deficiencies of CBC-MAC, but requires three keys. Iwata and Kurosawa proposed an improvement of XCBC and named the resulting algorithm One-Key CBC-MAC (OMAC) in their papers.[4][5] They later submitted OMAC1[6], a refinement of OMAC, and additional security analysis.[7] The OMAC algorithm reduces the amount of key material required for XCBC. CMAC is equivalent to OMAC1.



To generate an ℓ-bit CMAC tag (t) of a message (m) using a b-bit block cipher (E) and a secret key (k), one first generates two b-bit sub-keys ($k1$ and $k2$) using the following algorithm (this is equivalent to multiplication by x and x^2 in a finite field GF(2^b)). Let denote the standard left-shift operator and $\oplus$ denote exclusive or:

1. Calculate a temporary value $k_0 = E_k(0)$.
2. If msb($k0$) = 0, then $k1 = k0 \ll 1$, else $k1 = (k0 \ll 1) \oplus C$; where C is a certain constant that depends only on b. (Specifically, C is the non-leading coefficients of the lexicographically first irreducible degree-b binary polynomial with the minimal number of ones.)
3. If msb($k1$) = 0, then $k2 = k1 \ll 1$, else $k2 = (k1 \ll 1) \oplus C$.
4. Return keys ($k1$, $k2$) for the MAC generation process.

As a small example, suppose $b = 4$, $C = 0011_2$, and $k_0 = E_k(0) = 0101_2$. Then $k_1 = 1010_2$ and $k_2 = 0100 \oplus 0011 = 0111_2$.

The CMAC tag generation process is as follows:

1. Divide message into b-bit blocks $m = m_1 \| ... \| m_{n-1} \| m_n$ where m_1, ..., m_{n-1} are complete blocks. (The empty message is treated as 1 incomplete block.)
2. If m_n is a complete block then $m_n' = k_1 \oplus m_n$ else $m_n' = k_2 \oplus (m_n \| 10...0_2)$. 3. Let $c0 = 00…02$.
4. For $i = 1, ..., n-1$, calculate $c_i = E_k(c_{i-1} \oplus m_i)$.
5. $c_n = E_k(c_{n-1} \oplus m_n')$
6. Output $t = \text{msb}_t(c_n)$.

The verification process is as follows:

1. Use the above algorithm to generate the tag.
2. Check that the generated tag is equal to the received tag.

6). DIGITAL SIGNATURES

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

• It must verify the author and the date and time of thesignature

It must to authenticate the contents at the time of the signature • It must be verifiable by third parties, to resolve disputes Thus, the digital signature function includes the authentication function. A variety of approaches has been proposed forthe digital signature function. These approaches fall into two categories: direct andarbitrated.

Direct Digital Signature

Direct Digital Signatures involve the direct application of public-key algorithms involving only the communicating parties. A digital signature may be formed by encrypting the

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

Arbitrated Digital Signature

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

Using Conventional encryption

XA : M || E (Kxa ,[IDx || H (M)])

A Y : E(Kay ,[IDx || M || E (Kxa ,[IDx ||H(M))]) || T])

It is assumed that the sender X and the arbiter A share a secret key Kxa and that A and Y share secret key Kay. X constructs a message M and computes its hash value H(m) Then X transmits the message plus a signature to A. the signature consists of an identifier IDx of X plus the hash value, all encrypted using Kxa.

A decrypts the signature and checks the hash value to validate the message. Then A transmits a message to Y, encrypted with Kay.

The message includes IDx, the original message from X, the signature, and a timestamp.

Arbiter sees message

Problem : the arbiter could form an alliance with sender to deny a signed message, or with the receiver to forge the sender's signature.

Using Public Key Encryption

X : IDx ||E(PRx,[IDx|| E (PUy, E(PRx, M))]) A

A Y : E(PRa, [IDx ||E (PUy, E (PRx, M))|| T])

X double encrypts a message M first with X's private key,PRx, and then with Y's public key, PUy. This is a signed, secret version of the message. This signed message, together with X's identifier , is encrypted again with PRx and, together with IDx, is sent to A. The inner, double encrypted message is secure from the arbiter (and everyone else except Y)

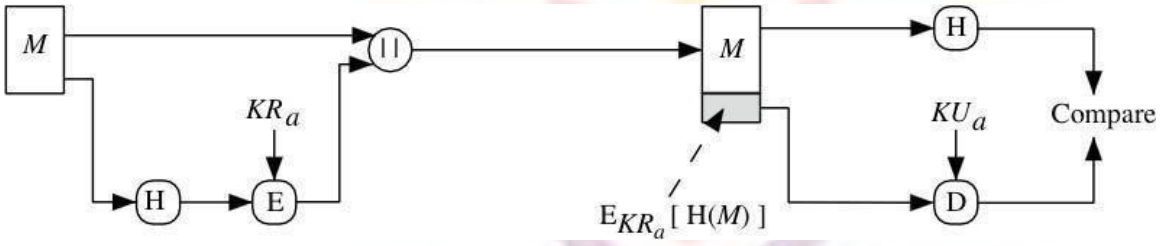
A can decrypt the outer encryption to assure that the message must have come from X (because only X has PRx). Then A transmits a message to Y, encrypted with PRa. The message includes IDx, the double encrypted message, and a timestamp.Arbiter does not see message

Digital Signature Standard (DSS)

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) and presents a new digital signature technique, the Digital Signature Algorithm (DSA). The DSS uses an algorithm that is designed to provide only the digital signature function and cannot be used for encryption or key exchange, unlike RSA.

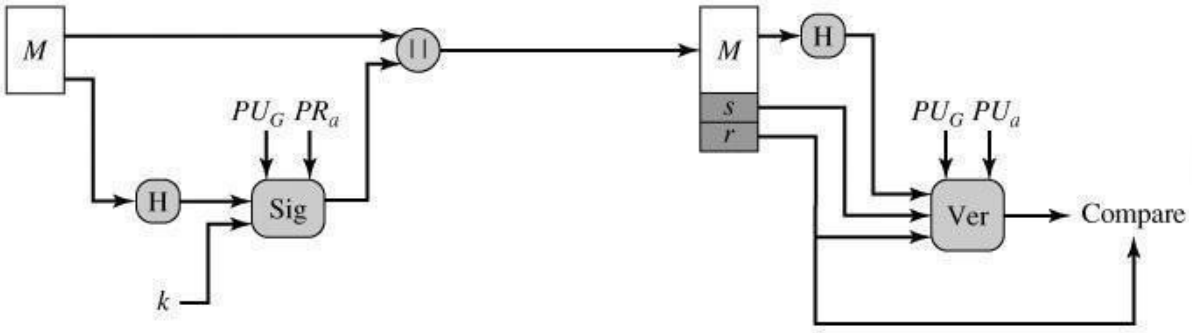
The RSA approach is shown below. The message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted.

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.

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 k 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 (PUG).The result is a signature consisting of two components, labeled s and r. 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



(b) DSS approach

depends on the global public key as well as the sender's public key (PUa), 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 r 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.

7). ELGAMAL DIGITAL SIGNATURE SCHEME

Before examining the NIST Digital Signature standard, it will be helpful to understand the ElGamal and Schnorr signature schemes. Recall from Chapter 10, that the ElGamal encryption scheme is designed to enable encryption by a user's public key with decryption by the user's private key. The ElGamal signature scheme involves the use of the private key for encryption and the public key for decryption [ELGA84, ELGA85].

Before proceeding, we need a result from number theory. Recall from Chapter 8 that for a prime number q, if a is a primitive root of q, then

$$\alpha, \alpha^2, \ldots, \alpha^{q-1}$$

are distinct (mod q). It can be shown that, if a is a primitive root of q, then

1. For any integer m, $\alpha^m \equiv 1 \pmod{q}$ if and only if $m \equiv 0 \pmod{q-1}$.
2. For any integers, i, j, $\alpha^i \equiv \alpha^j \pmod{q}$ if and only if $i \equiv j \pmod{q-1}$.

As with ElGamal encryption, the global elements of **ElGamal digital signature** are a prime number q and a, which is a primitive root of q. User A generates a private/public key pair as follows.

1. Generate a random integer XA, such that 1 6 XA<q - 1.
2. Compute $YA = aXA \bmod q$.
3. A's private key is XA; A's pubic key is {q, a, YA}.

To sign a message M, user A first computes the hash m = H(M), such that m is an integer in the range 0 <= m <= q - 1. A then forms a digital signature as follows.

1. Choose a random integer K such that 1 <= K <= q - 1 and gcd(K, q - 1) = 1. That is, K is relatively prime to q - 1.
2. Compute $S1 = aK \bmod q$. Note that this is the same as the computation of $C1$ for ElGamal encryption.
3. Compute K-1mod(q-1). That is, compute the inverse of K modulo q-1.
4. Compute $S2 = K\text{-}1(m - XAS1) \bmod (q - 1)$.
5. The signature consists of the pair ($S1$, $S2$).

Any user B can verify the signature as follows.

1. Compute $V1 = am \bmod q$.
2.

Compute $V2 = (YA)\ 1(S1) \bmod q$.

The signature is valid if $V1 = V2$. Let us demonstrate that this is so. Assume that the equality is true. Then we have

$\alpha^m \bmod q = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$	assume $V_1 = V_2$
$\alpha^m \bmod q = \alpha^{X_A S_1}\alpha^{KS_2} \bmod q$	substituting for Y_A and S_1
$\alpha^{m-X_A S_1} \bmod q = \alpha^{KS_2} \bmod q$	rearranging terms
$m - X_A S_1 \equiv KS_2 \bmod (q-1)$	property of primitive roots
$m - X_A S_1 \equiv KK^{-1}(m - X_A S_1) \bmod (q-1)$	substituting for S_2

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 8.3. We choose $a = 10$.

Alice generates a key pair as follows:

1. Alice chooses $XA = 16$.
2. Then $YA = aXA \bmod q = a16 \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is $\{q, a, YA\} = \{19, 104$. Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S1 = aK \bmod q = 105 \bmod 19 = 3$ (see Table 8.3).
3. $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$.
4. $S_2 = K^{-1}(m - X_A S_1) \bmod (q-1) = 11(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Bob can verify the signature as follows.

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$.
2. $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$.

Thus, the signature is valid.

8). SYMMETRIC KEY DISTRIBUTION USING SYMMETRIC ENCRYPTION

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the *key distribution technique*, a term that refers to the means of de liver- ing a key to two parties who wish to exchange data without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

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

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

The scale of the problem depends on the number of communicating pairs that must be supported. If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate. Thus, if there are N hosts, the number of required keys is $[N(N - 1)]/2$. If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication. Thus, a network may have hundreds of hosts but thousands of users and processes. Figure 14.1 illustrates the magnitude of the key distribution task for end-to-end encryption.1 A network using node-level encryption with 1000 nodes would conceivably need to distribute as many as half a million keys. If that same network supported 10,000 applications, then as many as 50 million keys may be required for application-level encryption.

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

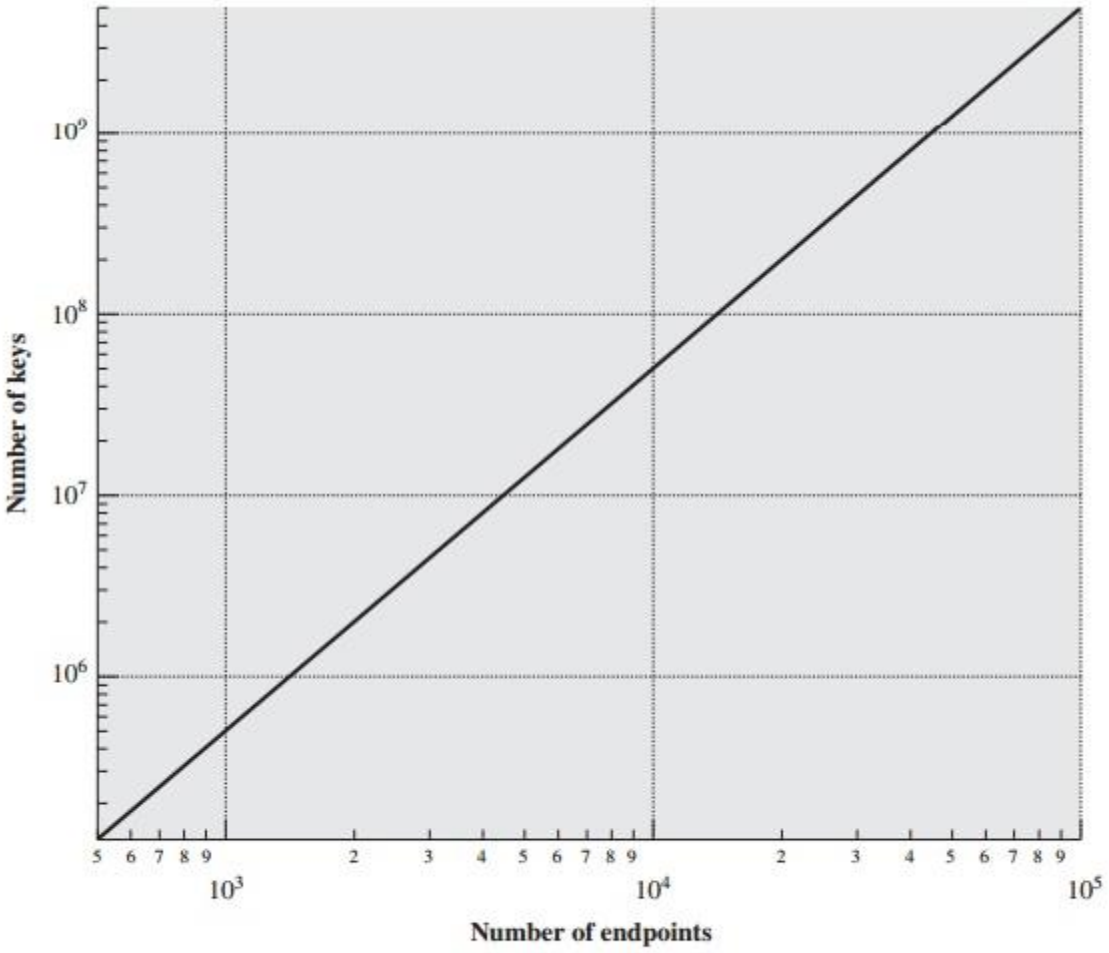


Figure 14.1 Number of Keys Required to Support Arbitrary Connections between Endpoints

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

The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used (Figure 14.2). Communication between
end systems is encrypted using a temporary key, often referred to as a **session key**. Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded. Each session
key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session
keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.

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

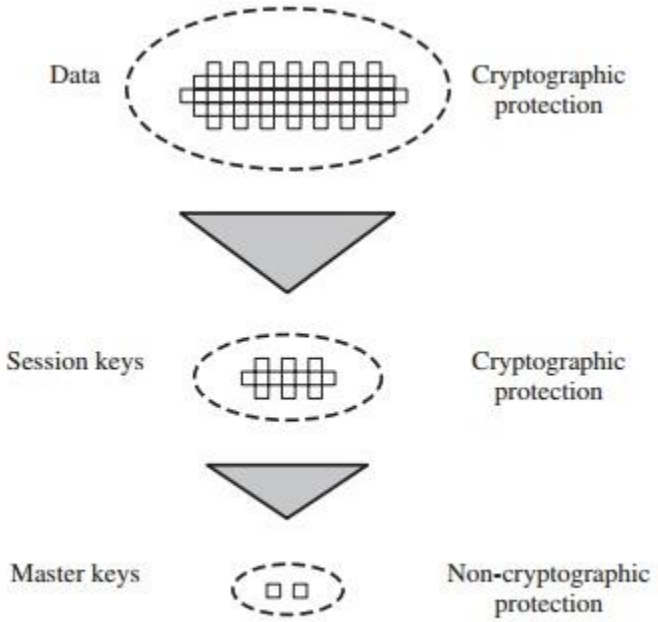


Figure 14.2 The Use of a Key Hierarchy

A Key Distribution Scenario

The key distribution concept can be deployed in a number of ways. A typical scenario is illustrated in Figure 14.3, which is based on a figure in [POPE79]. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

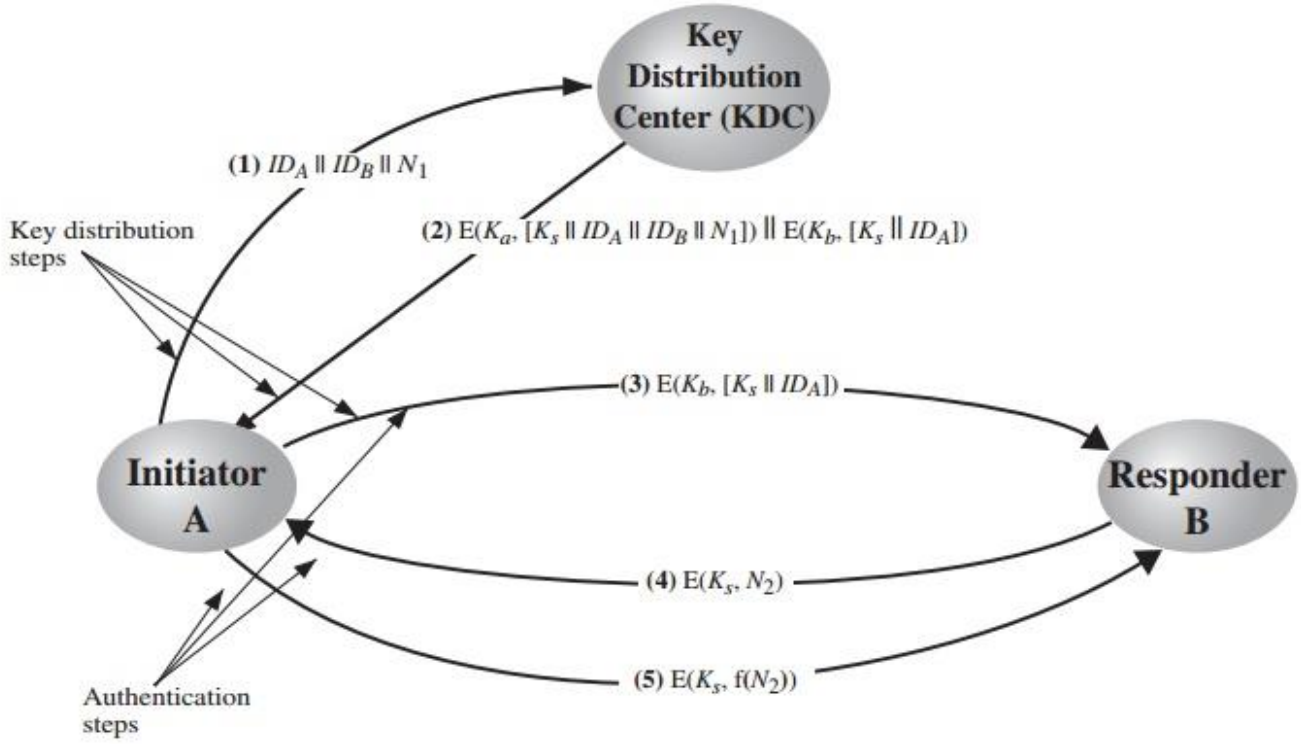


Figure 14.3 Key Distribution Scenario

Let us assume that 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, *Ka*, known only to itself and the KDC; similarly, B shares the master key *Kb* 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 time-stamp, 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 *Ka*. Thus, A is the only one who can success fully read the message, and A knows that it originated at the KDC. The message includes two ite ms intended for A:
   - The one-time session key, *Ks*, 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, *Ks*, to be used for the session
- An identifier of A (e.g., its network address), *IDA*

These last two items are encrypted with *Kb* (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.

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, $E(Kb,[Ks \| IDA])$. Because this information is encrypted with *Kb*, it is protected from eav esdrop- ping. B now knows the session key (*Ks*), knows that the other party is A (from *IDA*), and knows that the information originated at the KDC (because it is encrypted using *Kb*).

At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, $N2$, to A.
5. Also, using *Ks*, A responds with $f(N2)$, where f is a function that performs some transformation on $N2$ (e.g., adding one).

These steps assure B that the original message it received (step 3) was not a replay.

Note that the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

Hierarchical Key Control

It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so. As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. In this case, any one of the three KDCs involved can actually select the key. The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork.

A hierarchical scheme minimizes the effort involved in master key distribu- tion, because most master keys are those shared by a local KDC with its local enti- ties. Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

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.

A Transparent Key Control Scheme

The approach suggested in Figure 14.3 has many variations, one of which is described in this subsection. The scheme (Figure 14.4) is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users. The approach assumes that communication makes use of a connection-oriented end-to-end protocol, such as TCP. The noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

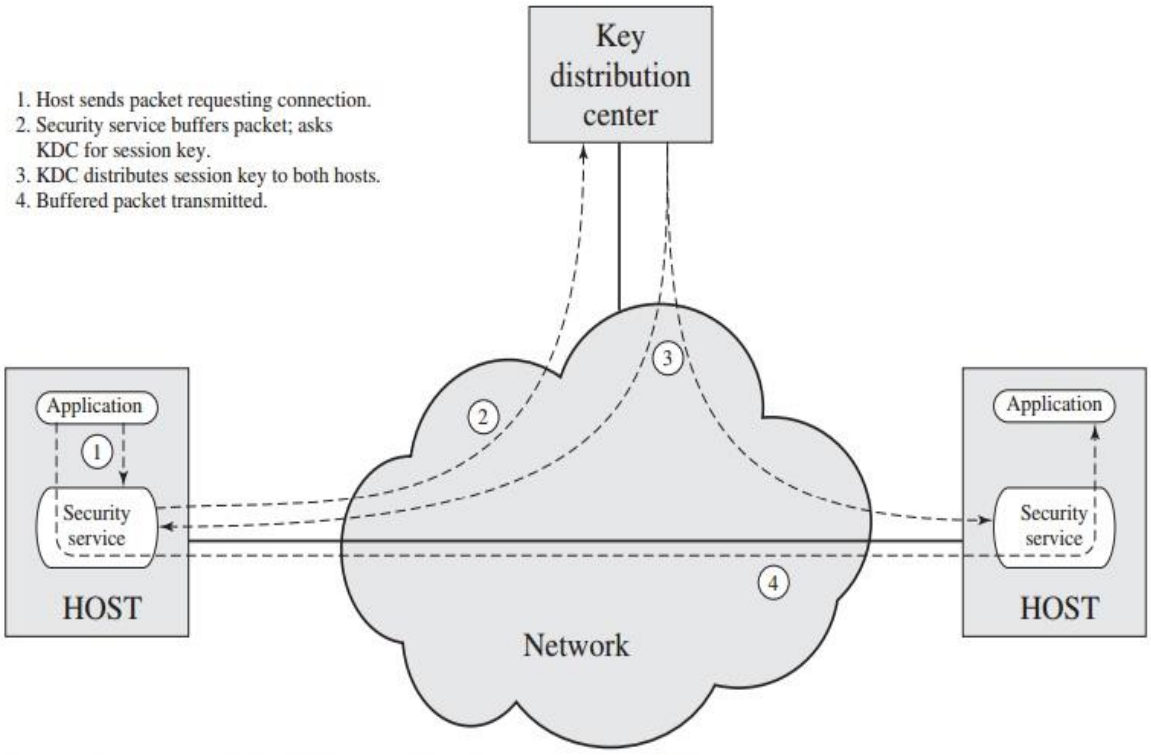


Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

The stepsinvolved in establishing a connection are shown in Figure 14.4. When one host wishes to set up a connection to another host, it transmits a connec-tion request packet (step 1). The SSM saves that packet and applies to the KDC for permission to establish the connection (step 2). The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM (step 3). The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4). All user data exchanged between the two end systems are encrypted by their respective SSMs using the one- time session key.

The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

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. Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

A decentralized approach requires that each end system be able to communi- cate in a secure manner with all potential partner end systems for purposes of ses- sion 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 (Figure 14.5).

1. A issues a request to B for a session key and includes a nonce, $N1$.
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($N1$), and another nonce, $N2$.
3. Using the new session key, A returns f($N2$) to B.

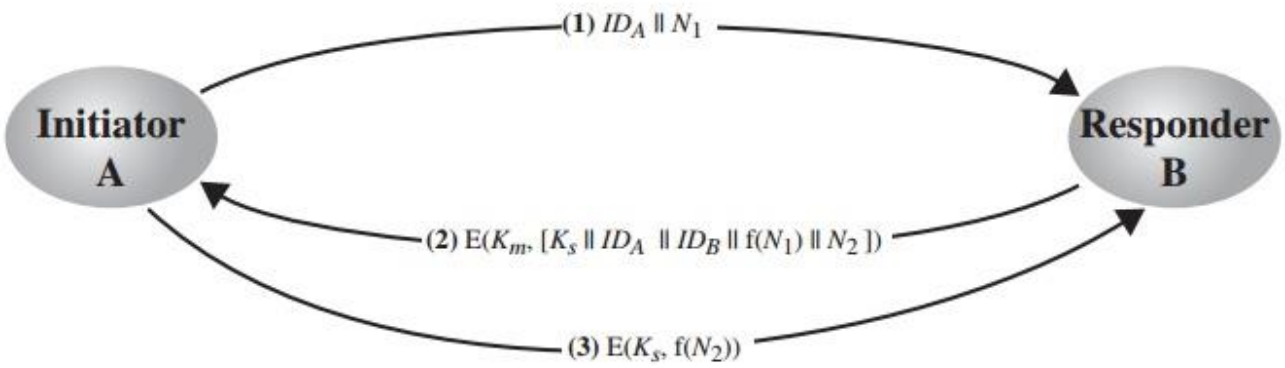


Figure 14.5 Decentralized Key Distribution

Thus, although each node must maintain at most $(n - 1)$ master keys, as many session keys as required may be generated and used. Because the messages trans- ferred using the master key are short, cryptanalysis is difficult. As before, session keys are used for only a limited time to protect them.

Controlling Key Usage

The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed. It also may be desirable to impose some control on the way in which automatically distributed keys are used. For example, in addition to separating mas- ter keys from session keys, we may wish to define different types of session keys on the basis of use, such as

- Data-encrypting key, for general communication across a network
- PIN-encrypting key, for personal identification numbers (PINs) used in elec- tronic funds transfer and point-of-sale applications
- File-encrypting key, for encrypting files stored in publicly accessible locations

To illustrate the value of separating keys by type, consider the risk that a master key is imported as a data-encrypting key into a device. Normally, the mas- ter key is

physically secured within the cryptographic hardware of the key distrib- ution center and of the end systems. Session keys encrypted with this master key are available to application programs, as are the data encrypted with such session keys. However, if a master key is treated as a session key, it may be possible for an unauthorized application to obtain plaintext of session keys encrypted with that master key.

Thus, it may be desirable to institute controls in systems that limit the ways in which keys are used, based on characteristics associated with those keys. One simple plan is to associate a tag with each key ([JONE82]; see also [DAVI89]). The proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key. That is, the eight non-key bits ordinarily reserved for parity checking form the key tag. The bits have the following interpretation:

- One bit indicates whether the key is a session key or a master key.
- One bit indicates whether the key can be used for encryption.
- One bit indicates whether the key can be used for decryption.
- The remaining bits are spares for future use.

Because the tag is embedded in the key, it is encrypted along with the key when that`key i s distributed, thus providing protection. The drawbacks of this scheme are

1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

A more flexible scheme, referred to as the control vector, is described in [MATY91a and b]. In this scheme, each session key has an associated control vector

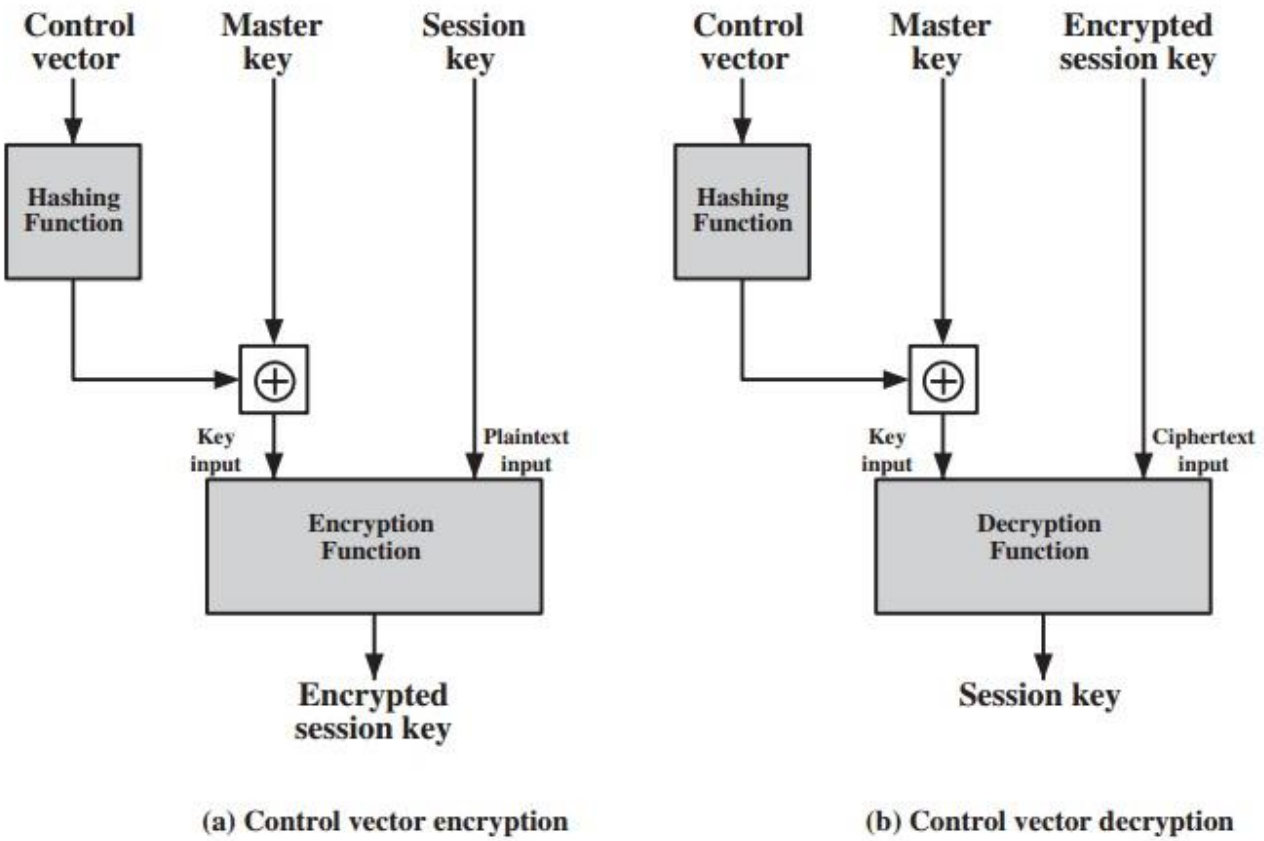


Figure 14.6 Control Vector Encryption and Decryption

consisting of a number of fields that specify the uses and restrictions for that session key. The length of the control vector may vary.

The control vector is cryptographically coupled with the key at the time of key generation at the KDC. The coupling and decoupling processes are illustrated in Figure 14.6. As a first step, the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length. Hash functions are discussed in detail in Chapter 11. In essence, a hash function maps values from a larger range into a smaller range with a reasonably uniform spread. Thus, for example, if numbers in the range 1 to 100 are hashed into numbers in th e range 1 to 10, approximately 10% of the source values should map into each of the targ et values.

The hash value is then XORed with the master key to produce an output that is used as the key input for encrypting the session key. Thus,

Hash value = H = h(CV)

Key input = K_m Ⓧ H

Ciphertext = $E([K_m \otimes H], K_s)$

where K_m is the master key and K_s is the session key. The session key is recovered in plaintext by the reverse operation:

$D([K_m \otimes H], E([K_m \otimes H], K_s))$

When a session key is delivered to a user from the KDC, it is accompanied by the control vector in clear form. The session key can be recovered only by using both the master key that the user shares with the KDC and the control vector. Thus, the linkage between the session key and its control vector is maintained.

Use of the control vector has two advantages over use of an 8-bit tag. First, there is no restriction on length of the control vector, which enables arbitrarily com- plex controls to be imposed on key use. Second, the control vector is available in clear form at all stages of operation. Thus, control of key use can be exercised in multiple locations.

9). SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

Because of the inefficiency of public key cryptosystems, they are almost never used for the direct encryption of sizable block of data, but are limited to relatively small blocks. One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution. We see many specific examples of this in Part Five. Here, we discuss general principles and typical approaches.

Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle [MERK79], as illustrated in Figure 14.7. If A wishes to communicate with B, the following procedure is employed:

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.

A and B can now securely communicate using conventional encryption and the session key K_s. At the completion of the exchange, both A and B discard K_s.

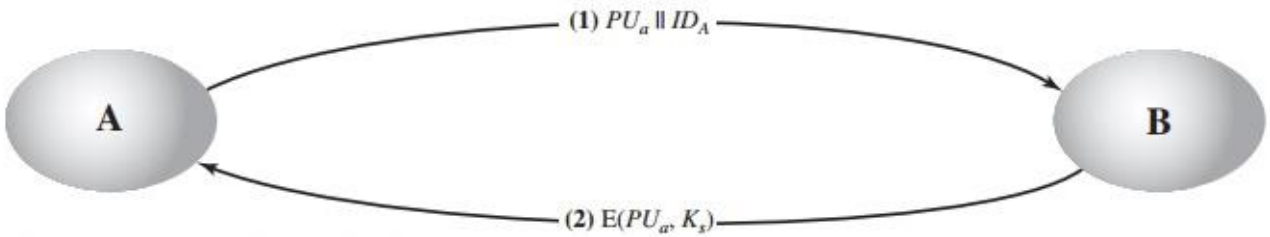


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.

The protocol depicted in Figure 14.7 is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message (see Figure 1.3c). Such an attack is known as a **man-in-the-middle attack** [RIVE84]. In this case, if an adversary, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion without being detected.

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message intended for B consisting of PU_a and an identifier of A, ID_A.
2. E intercepts the message, creates its own public/private key pair $\{PU_e, PR_e\}$ and transmits $PU_e \| ID_A$ to B.
3. B generates a secret key, K_s, and transmits $E(PU_e, K_s)$.
4. E intercepts the message and learns K_s by computing $D(PR_e, E(PU_e, K_s))$.
5. E transmits $E(PU_a, K_s)$ to A.

The result is that both A and B know K_s and are unaware that K_s has also been revealed to E. A and B can now exchange messages using K_s. E no longer actively interferes with the communications channel but simply eavesdrops. Knowing K_s, E can decrypt all messages, and both A and B are unaware of the problem. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication

Figure 14.8, based on an approach suggested in [NEED78], provides protection against both active and passive attacks. We begin at a point when it i

s assumed that
A and B have exchanged public keys by one of the schemes described subsequently in this chapter. Then the following steps occur.

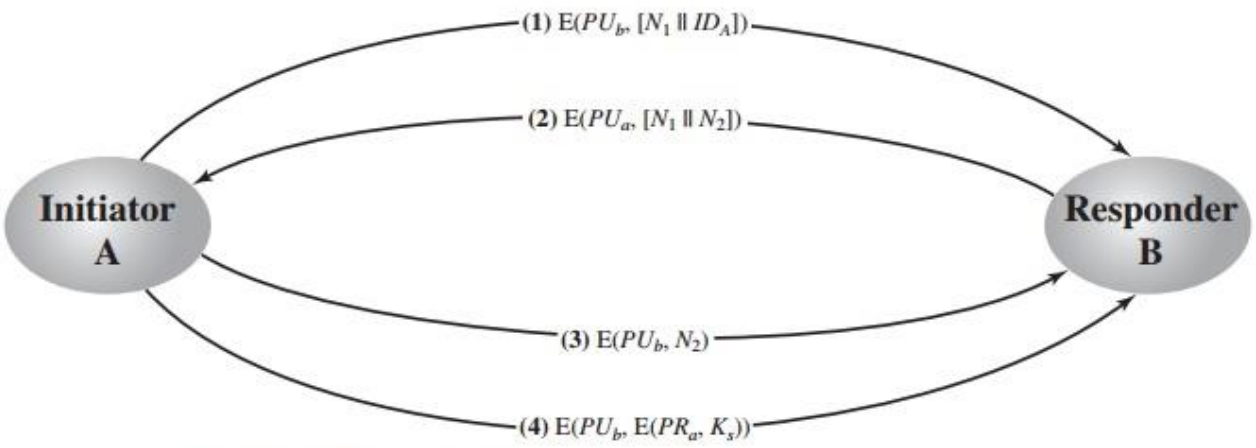


Figure 14.8 Public-Key Distribution of Secret Keys

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.

B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as ell as a new nonce generated by B (N_2). Because only B could have (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.

2. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

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.

3. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.
4. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available

dynamic directory of public keys. Maintenance and distribution of the public direc- tory would have to be the responsibility of some trusted entity or organization (Figure 14.10). Such a scheme would include the following elements:

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

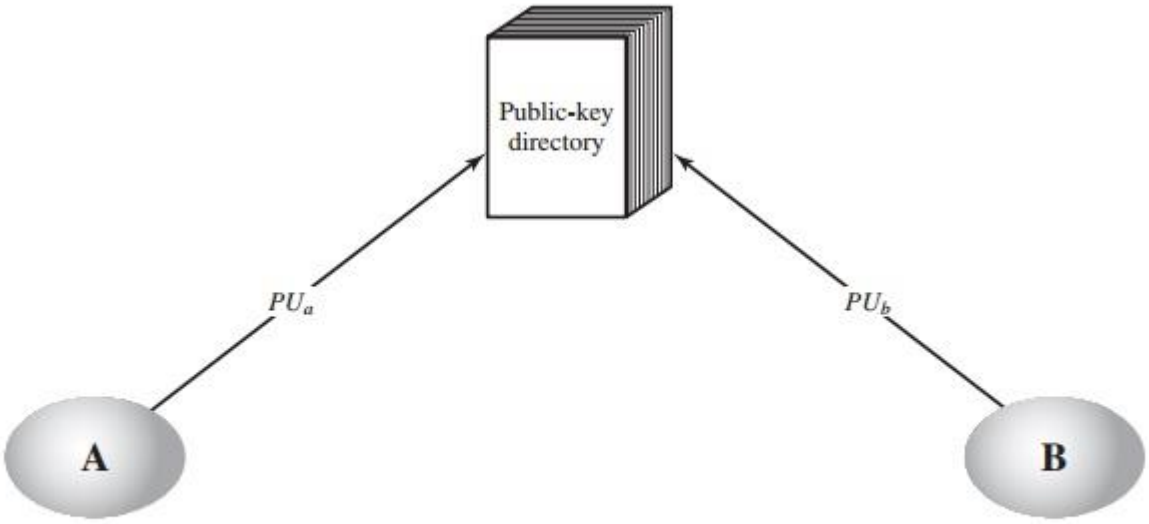


Figure 14.10 Public-Key Publication

4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

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

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure 14.11, which is based on a figure in [POPE79]. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with

only the authority knowing the corresponding private key. The following steps (matched by number to Figure 14.11) occur.

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, *PUb*, 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

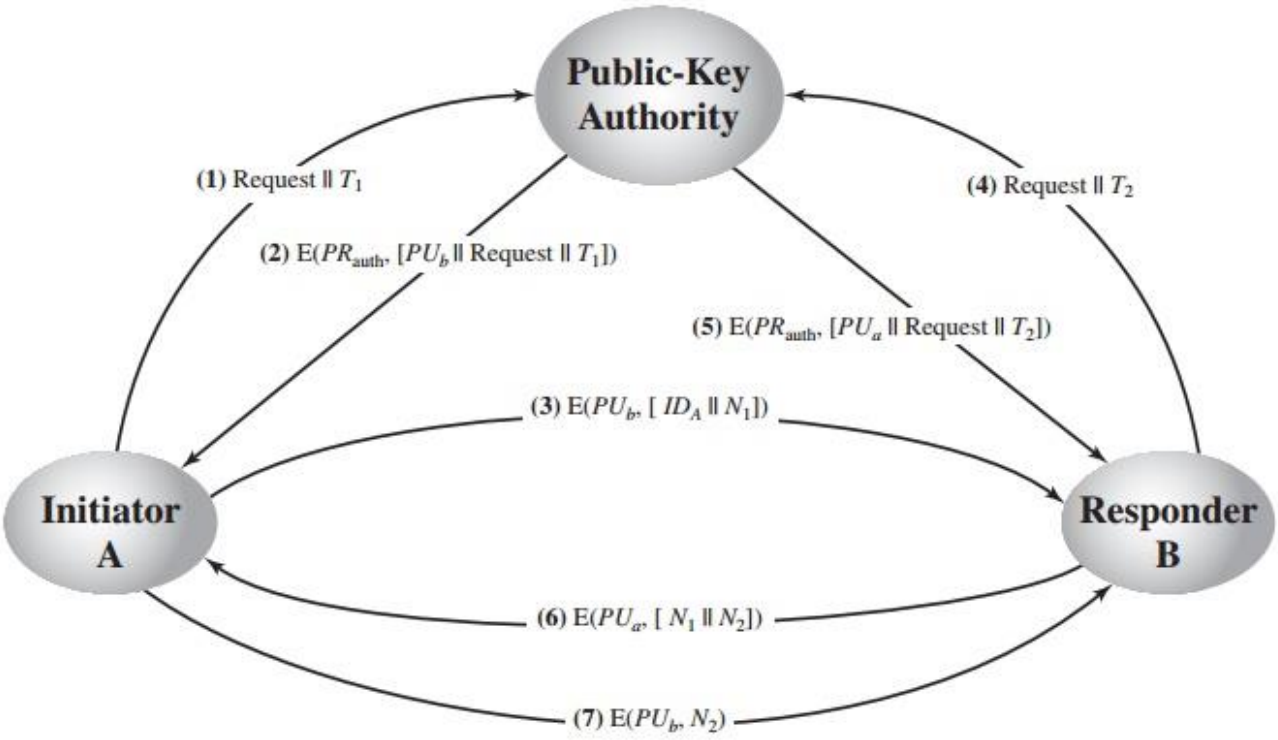


Figure 14.11 Public-Key Distribution Scenario

   - 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 (*IDA*) 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 PUa 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.

6.A returns *N*2, which is encrypted using B's public key, to assure B that its cor- respondent is A.

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

Public-Key Certificates

The scenario of Figure 14.11 is attractive, yet it has some drawbacks. The public-key
authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vul- nerable to tampering.

An alternative approach, first suggested by Kohnfelder [KOHN78], is to use **certificates** that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner and obtain a cer- tificate. The user can then publish the certificate. Anyone needing this user's pub- lic key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

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

These requirements are satisfied by the original proposal in [KOHN78]. Denning [DENN83] added the following additional requirement:

4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure 14.12. Each participant applies to the certificate authority, supplying a public

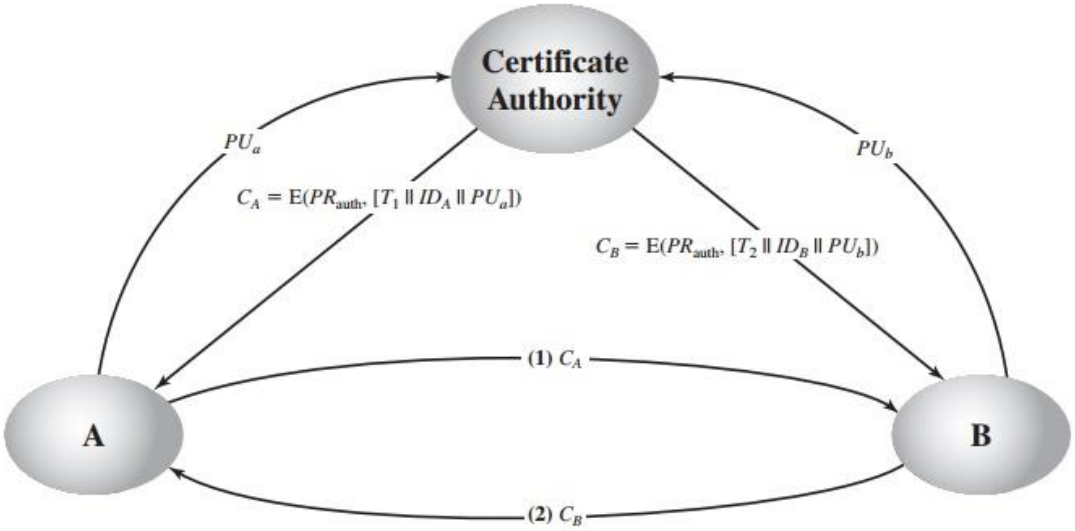


Figure 14.12 Exchange of Public-Key Certificates

Application must be in person or by some form of secure authenticated communi- cation. For participant A, the authority provides a certificate of the form

$CA = E(PR\text{auth}, [T \| IDA \| PUa])$

where PRauth is the private key used by the authority and T is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the cer- tificate as follows:

$D(PU\text{auth}, CA) = D(PU\text{auth}, E(PR\text{auth}, [T \| IDA \| PUa])) = (T \| IDA \| PUa)$

The recipient uses the authority's public key, PUauth, to decrypt the certifi- cate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements *IDA* and *PUa* provide the recipient with the name and public key of the certificate's holder. The timestamp *T* validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A gen- erates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired.

One scheme has become universally accepted for formatting public-key cer- tificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, transport layer security (TLS), and S/MIME.

10). KERBEROS

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

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

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

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

Reliable: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

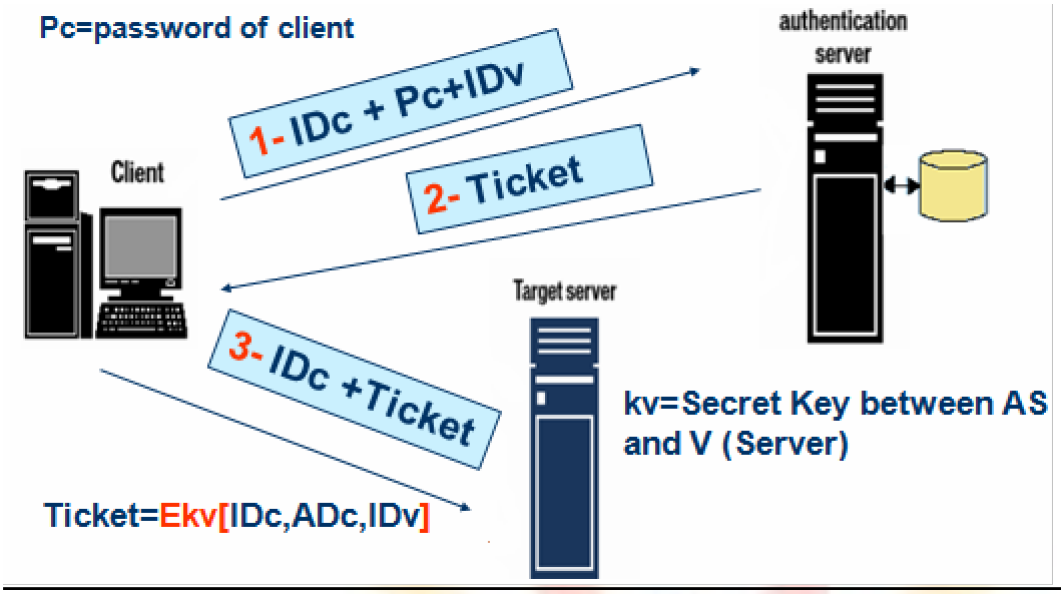
Transparent: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

Scalable: The system should be capable of supporting large numbers of clients and serv Two versions of Kerberos are in common use: Version 4 is most widely used version. Version 5 corrects some of the security deficiencies of Version 4. Version 5 has been issued as a draft Internet Standard (RFC 1510)

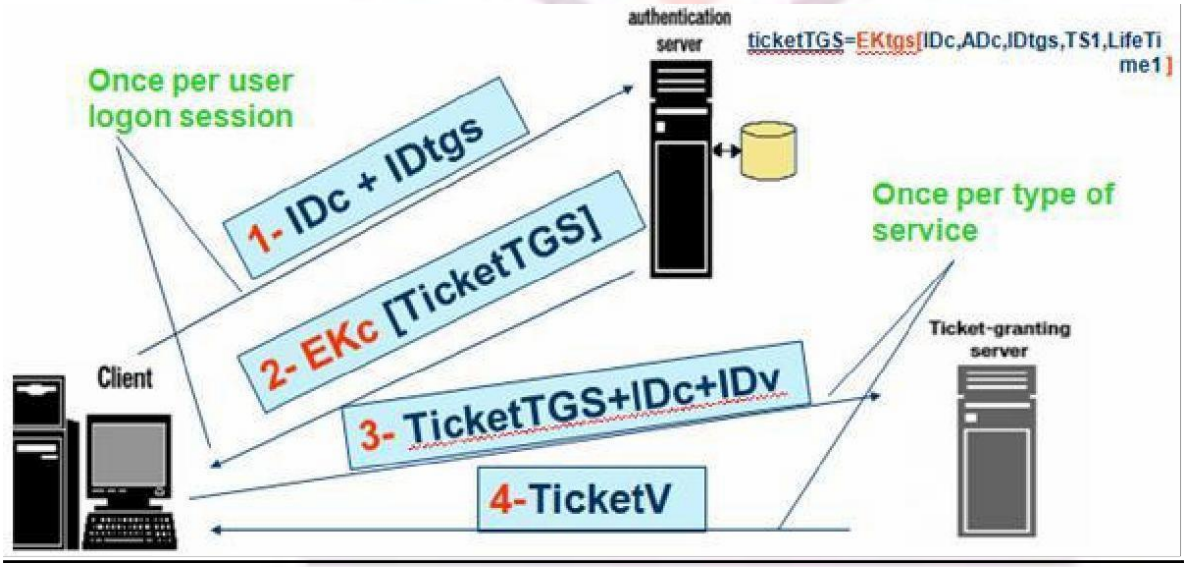
ers. This suggests a modular, distributed architecture

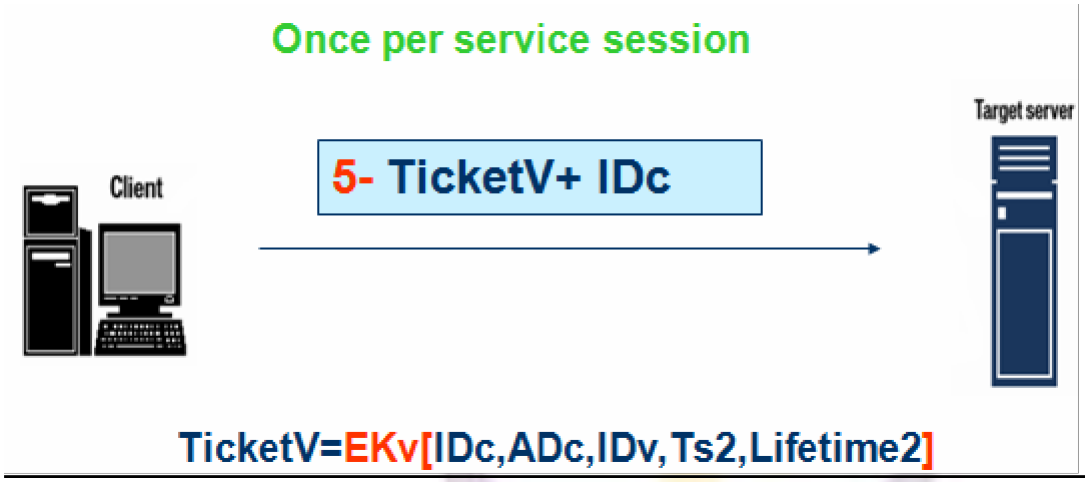
KERBEROS VERSION 4

1.) **SIMPLE DIALOGUE**:



MORE SECURE DIALOGUE





(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

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

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

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

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

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

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

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

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

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

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

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

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

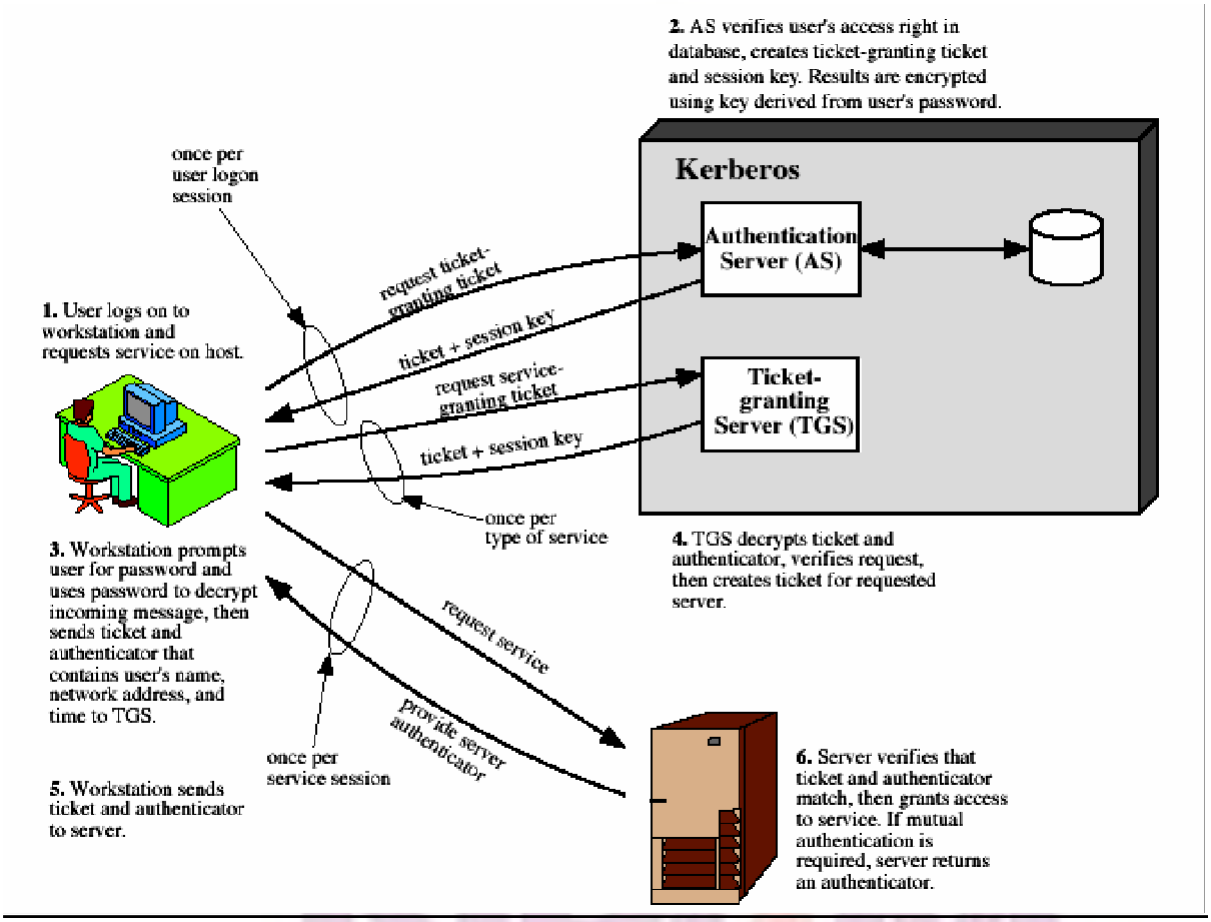
$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Client/Server Authentication Exchange to obtain service

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

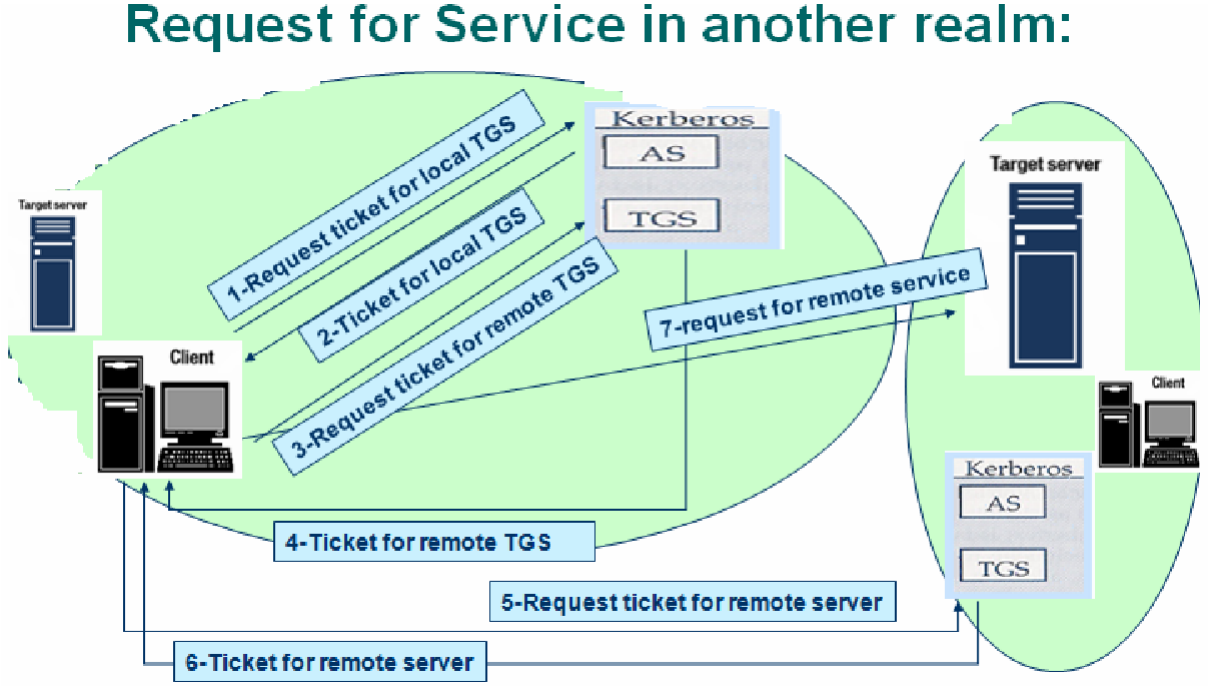
Message (1) includes a timestamp, so that the AS knows that the message is timely. Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the TGS and to learn its expiration time. Note that the ticket does not prove anyone's identity but is a way to distribute keys securely. It is the authenticator that proves the

client's identity. Because the authenticator can be used only once and has a short lifetime, the threat of an opponent stealing both the ticket and the authenticator for presentation later is countered. C then sends the TGS a message that includes the ticket plus the ID of the requested service (message 3). The reply from the TGS, in message (4), follows the form of message (2). C now has a reusable service- granting ticket for V. When C presents this ticket, as shown in message (5), it also sends an authenticator



Kerberos Realms A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers is referred to as a Kerberos realm. A Kerberos realm is a set of managed nodes that share the same Kerberos database, and are part of the same administrative domain. If have multiple realms, their Kerberos servers must share keys and trust each other.

The following figure shows the authentication messages where service is being requested from another domain. The ticket presented to the remote server indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request. One problem presented by the foregoing approach is that it does not scale well to many realms, as each pair of realms need to share akey.



Advantages of Kerberos:

User's passwords are never sent across the network,

encrypted or in plain text Secret keys are *only* passed acrossthe

network in encrypted form

Client and server systems

mutually authenticate limits the It duration of their users' authentication.

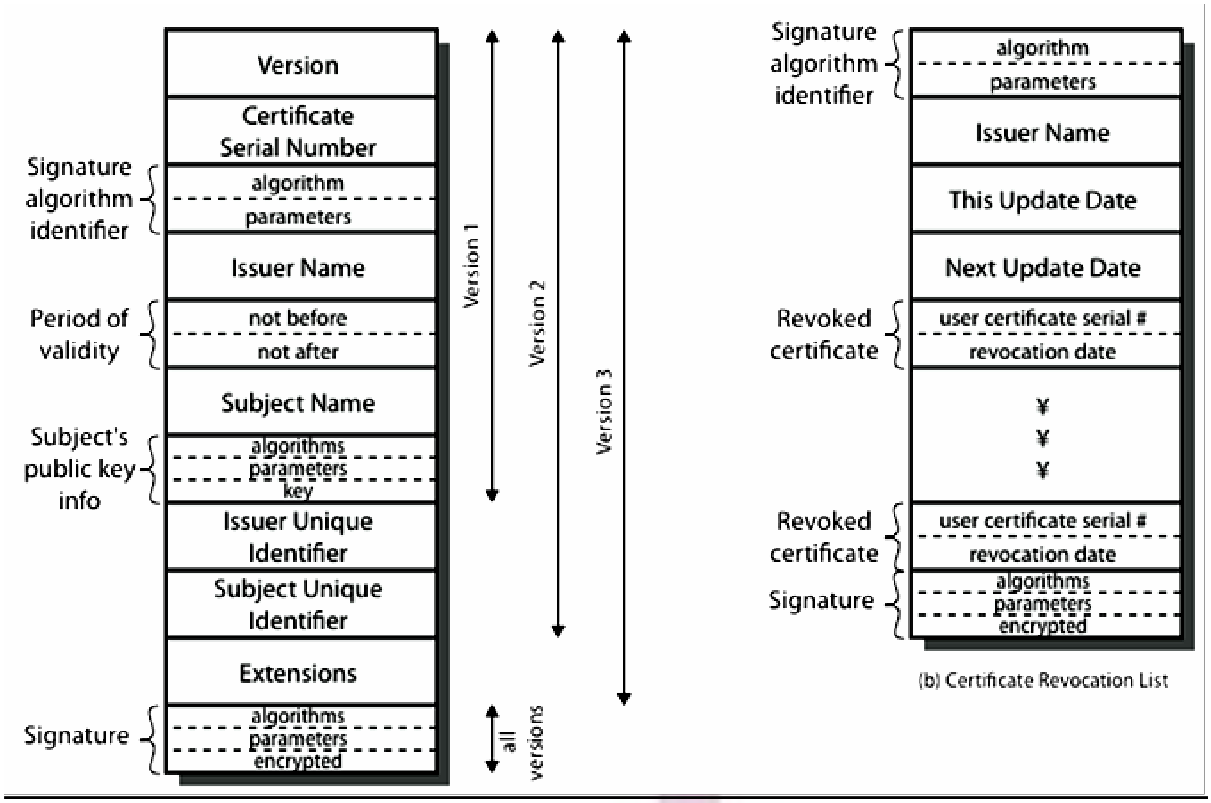
Authentications are **reusable** and **durable**

Kerberos has been scrutinized by many of the top programmers, cryptologists and security experts in the industry

11). X.509 AUTHENTICATION SERVICES

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

The general format of a certificate is shown above, whichincludes the following elements :version 1, 2, or 3serial number (unique within CA) identifying certificate signature algorithm identifier issuer X.500 name (CA) period of validity (from - to dates) subject X.500 name (name of owner)subject public-key info (algorithm,parameters, key) issuer unique identifier (v2+)



The standard uses the following notation to define a certificate:

CA<<A>> = CA {V, SN, AI, CA, TA, A, Ap}Where Y<<X>>= the certificate of user X issued by certification authority YY {I} == the signing of I by Y. It consists of I wann encrypted

hash code appended User certificates generated by a CA have the following characteristics:

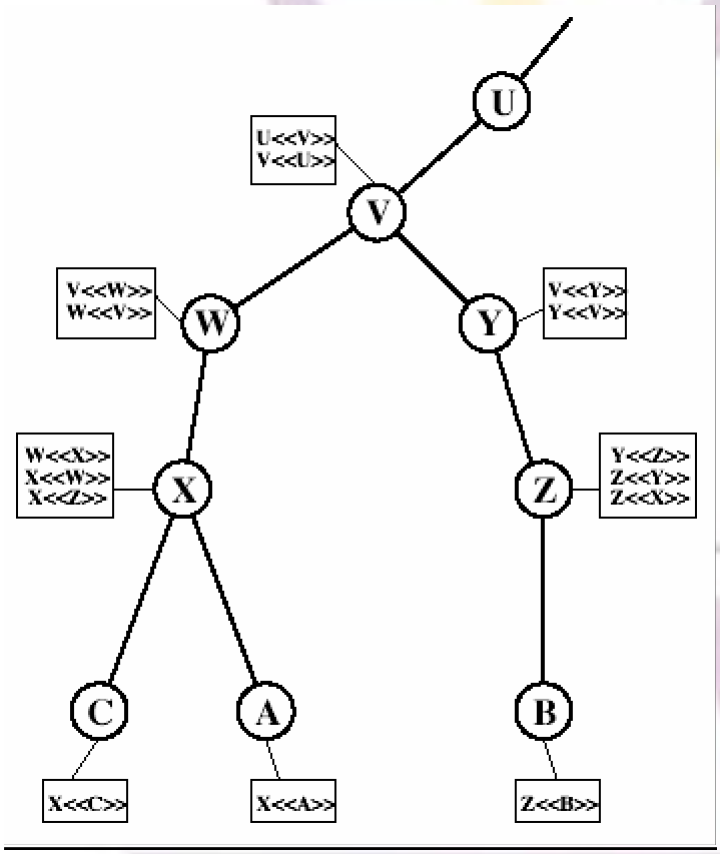
Any user with CA's public key can verify the user public key that was certified No party other than the CA can modify the certificate without being detected because they cannot be forged, certificates can be placed in a public directory

Scenario: Obtaining a User Certificate If both users share a common CA then they are assumed to know its public key. Otherwise CA's must form a hierarchy and use certificates linking members of hierarchy to validate other CA's. Each CA has certificates for clients (forward) and parent (backward). Each client trusts parents certificates. Itenables verification of any certificate from one CA by users of all other CAs in

hierarchy. A has obtained a certificate from the CA X1. B has obtained a certificate from the CA X2. A can read the B's certificate but cannot verify it. In order to solve the problem ,the Solution: **X1<<X2> X2<<B>>.** A obtain the certificate of X2 signed byX1 from directory.ain X2's public key. A goes back to directory and obtain the certificate of B signed by X2. obtain B's public key securely. The directory entry for each CA includes two types of certificates: Forward certificates: Certificates of X generated by other Cas Reverse certificates: Certificates generated by X that are the certificates of other Cas

X.509 CA Hierarchy

A acquires B certificate using chain:X<<W>>W<<V>>V<<Y>>Y<<Z>> Z<<B>>
B acquires A certificate using chain:Z<<Y>>Y<<V>>V<<W>>W<<X>> X<<A>>



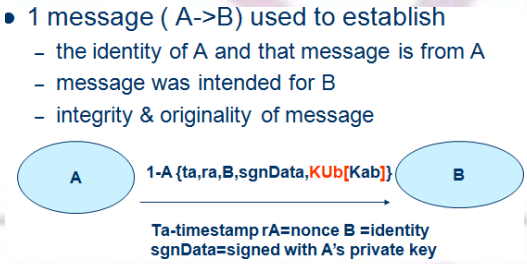
just before the expiration of the old one. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:The user's private key is assumed tobe compromised. The user is no longer certified by this CA.The CA's certificate is assumed to be compromised.Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users

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

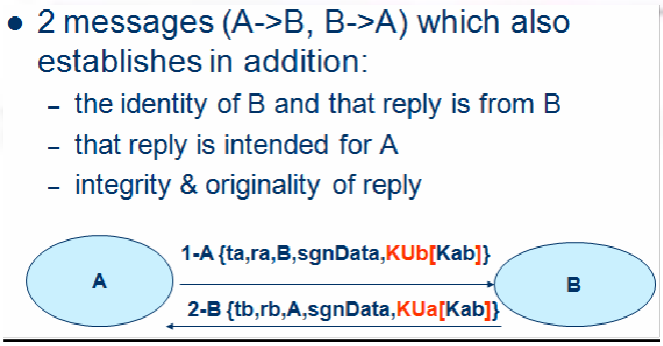
AUTHENTICATION PROCEDURES

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications. All these procedures make use of public-key signatures. It is assumed that the two parties know each other's public key, either by obtaining each other's certificates from the directory or because the certificate is included in the initial message from each side. 1. One-Way Authentication: One way authentication involves a single transfer of information from one user (A) to another (B), and establishes the details shown above.

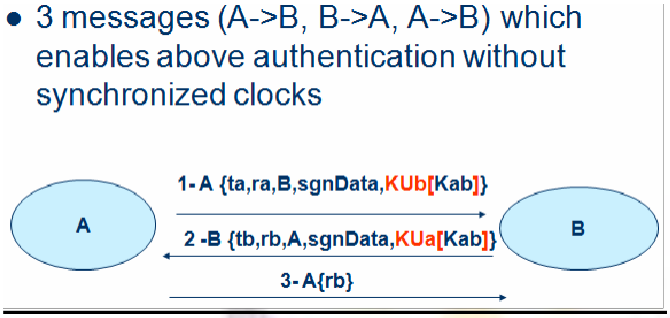
Note that only the identity of the initiating entity is verified in this process, not that of the responding entity. At a minimum, the message includes a timestamp ,a nonce, and the identity of B and is signed with A's private key. The message may also include information to be conveyed, such as a session key for B.



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



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



12). PUBLIC-KEY INFRASTRUCTURE

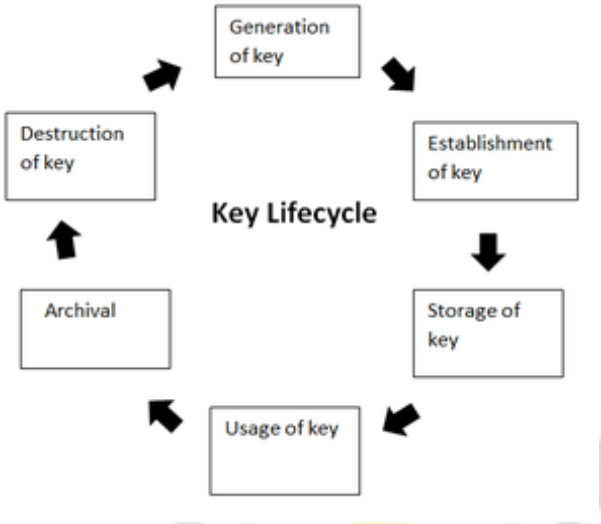
Public key infrastructure or PKI is the governing body behind issuing digital certificates. It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them

Managing Keys in the Cryptosystem:

The security of a cryptosystem relies on its keys. Thus, it is important that we have a solid key management system in place. The 3 main areas of key management are as follows:

- A cryptographic key is a piece of data that must be managed by secure administration.
- It involves managing the key life cycle which is as follows:



- Public key management further requires:
  - **Keeping the private key secret:** Only the owner of a private key is authorized to use a private key. It should thus remain out of reach of any other person.
  - **Assuring the public key:** Public keys are in the open domain and can be publicly accessed. When this extent of public accessibility, it becomes hard to know if a key is correct and what it will be used for. The purpose of a public key must be explicitly defined.

PKI or public key infrastructure aims at achieving the assurance of public key.

Public Key Infrastructure:

Public key infrastructure affirms the usage of a public key. PKI identifies a public key along with its purpose. It usually consists of the following components:

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system

Working on a PKI:

Let us understand the working of PKI in steps.

- **PKI and Encryption:** The root of PKI involves the use of cryptography and encryption techniques. Both symmetric and asymmetric encryption uses a public key. The challenge here is – "how do you know that the public key belongs to the right person or to the person you think it belongs to?". There is always a risk of MITM(Man in the middle). This issue is resolved by a PKI using digital certificates. It gives identities to keys in order to make the verification of owners easy and accurate.
- **Public Key Certificate or Digital Certificate:** Digital certificates are issued to people and electronic systems to uniquely identify them in the digital world. Here are a few noteworthy things about a digital certificate. Digital certificates are also called X.509 certificates. This is because they are based on the ITU standard X.509.
  - The Certification Authority (CA) stores the public key of a user along with other information about the client in the digital certificate. The information is signed and a digital signature is also included in the certificate.

- The affirmation for the public key then thus be retrieved by validating the signature using the public key of the Certification Authority.

- **Certifying Authorities:** A CA issues and verifies certificates. This authority makes sure that the information in a certificate is real and correct and it also digitally signs the certificate. A CA or ***Certifying Authority performs these basic roles***:
  - Generates the key pairs – This key pair generated by the CA can be either independent or in collaboration with the client.
  - Issuing of the digital certificates – When the client successfully provides the right details about his identity, the CA issues a certificate to the client. Then CA further signs this certificate digitally so that no changes can be made to the information.
  - Publishing of certificates – The CA publishes the certificates so that the users can find them. They can do this by either publishing them in an electronic telephone directory or by sending them out to other people.
  - Verification of certificate – CA gives a public key that helps in verifying if the access attempt is authorized or not.
  - Revocation – In case of suspicious behavior of a client or loss of trust in them, the CA has the power to revoke the digital certificate.

Classes of a Digital Certificate:

A digital certificate can be divided into four broad categories. These are :

- Class 1: These can be obtained by only providing the email address.
- Class 2: These need more personal information.
- Class 3: This first checks the identity of the person making a request.
- Class 4: They are used by organizations and governments.

Process of creation of certificate:

The creation of a certificate takes place as follows:

- Private and public keys are created.
- CA requests identifying attributes of the owner of a private key.
- Public key and attributes are encoded into a CSR or Certificate Signing Request.
- Key owner signs that CSR to prove the possession of a private key.
- CA signs the certificate after validation.

Creation of Trust layers among CA Hierarchies:

Each CA has its own certificate. Thus, trust is built hierarchically where one CA issues certificates to other CAs. Moreover, there is a root certificate that is self-signed. For a root CA, the issuer and the subject are not two separate parties but a single party.

Security of Root CA:

As you saw above, the ultimate authority is the root CA. Hence, the security of root CA is of huge importance. If the private key of a root CA is not taken care of, then it might turn into a catastrophe. This is because anyone disguised as the root CA can then issue certificates. To meet security standards, a root CA should be offline 99.9% of the time. However, it does need to come online to create public and private keys and to issue new certificates. Ideally, these activities should be performed 2-4 times a year.

Use of PKI in Today's Digital Age:

Today, there are an enormous number of applications that need require authentication. Certifications are needed at millions of places. This can not be done without a Public key infrastructure. The importance of PKI, depending on the use case and needs, has evolved over time. Here is a part of that track.

- For the very first time during the period of 1995 to 2002, the use of PKI was limited to the most important and high-value certificates. This included the certificates of eCommerce websites that enabled them to display the lock icon in the search bar. The goal was to make consumers confident about the security and authenticity of various websites.
- The second episode of PKI emerged around 2003 to 2010 when enterprises came into the picture. It was at this time that employees received laptops and the use of mobile phones was rising. Thus, employees needed access to the organization's assets even outside the office. That is when the use of PKI looked like the best way for authentication.
- The third phase started in 2011 and is continuing to date. With the advent of new technologies like IoT(Internet of Things) and need the to scale PKI, the use, as well as the challenges in using PKI, have increased tremendously. Today, millions of certificates are issued to authenticate mobile workforces. However, managing this huge number of certificates is quite challenging.
- S/MIME, Document Signing, code or app signing also uses PKI.

Challenges that a PKI Solves:

PKI owes its popularity to the various problems its solves. Some use cases of PKI are:

- Securing web browsers and communicating networks by SSL/TLS certifications.
- Maintaining Access Rights over Intranets and VPNs.
- Data Encryption
- Digitally Signed Software
- Wi-fi Access Without Passwords

Other than these, one of the most important use cases of PKI is based around IoT(Internet of Things). Here are two industries that are using PKI for IoT devices:

- **Auto Manufacturers:** Cars these days have features like GPS, call for services, assistants, etc. These require communication paths where a lot of data is passed. Making these connections secure is very important to avoid malicious parties hacking into the cars. This is where PKI comes in.
- **Medical device Manufacturers:** Devices like surgical robots require high security. Also, FDA mandates that any next-generation medical device must be updatable so that bugs can be removed and security issues can be dealt with. PKI is used to issues certificates to such devices.

Disadvantages of PKI:

- **Speed:** Since PKI uses super complex algorithms to create a secure key pair. So it eventually slows down the process and data transfer.

- **Private Key Compromise:** Even though PKI can't be hacked very easily but a private key can be hacked by a professional hacker, since PKI uses Public and Private key to encrypt and decrypt data so with user's private key in hand and public key which is easily available the information can be decrypted easily.

Descriptive Questions:

(a) 2 Marks Questions

1. **List out the applications of the public key cryptosystems.**
   - Encryption / decryption
   - Digital signature

- Key exchange

2. List out the different techniques of distributing the public key.

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

3. What is meant by Message Authentication?

Message Authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion,deletion, or replay) and that the purported identity of the sender is valid.

4. Define the classes of message authentication function.

- Message encryption
- Message Authentication Code (MAC)
- Hash function

5. Define Hash function .

A function that maps a message of any length into a fixed length hash value, which serves as the authenticator

6. Differentiate Message Authentication Code and Hash function.

In MAC, a public function of the message and a secret key are used to produce a fixedlength authenticator.

A hash function accepts a variable size message as input and produces a fixed size output(hash code) which is similar to MAC. But hash code does not use a key.

7. Define one way property, weak collision resistance and strong collision resistance of hashfunction.

For any given value h, it is computationally infeasible to find x such that H(x) = h – oneway property.

For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x) –weak collision resistance.

It is computationally infeasible to find any pair (x, y) such that H(x) = H(y) – strongcollision property.

8. What you meant by MAC?

MAC is Message Authentication Code. It is a function of message and secret key whichproduce a fixed length value called as MAC.

T=MAC(K,M)

where M is a variable-length message, K is a secret key shared only by sender and receiver, andMAC(K,M) is the fixed-length authenticator.

9. List out the attack on MAC.

- Brute-force attacks
- Cryptanalysis.

10. Define Digital signature.

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message

11. What are the properties of Digital Signature?

The digital signature must have the following properties:
It must verify the author and the date and time of the signature. It must authenticate the contents at thetime of the signature. It must be verifiable by third parties, to resolve disputes

12. List out the attacks related to Digital Signature.

Key-only attack:

- Known message attack
- Generic chosen message attack
- Directed chosen message attack
- Adaptive chosen message attack

13. Mention the signature function in DSS ?

The hash **function** used in the **DSS** standard is specified in the Secure Hash Standard (SHS), whichare the specifications for the Secure Hash Algorithm (SHA).

14. Define Universal forgery

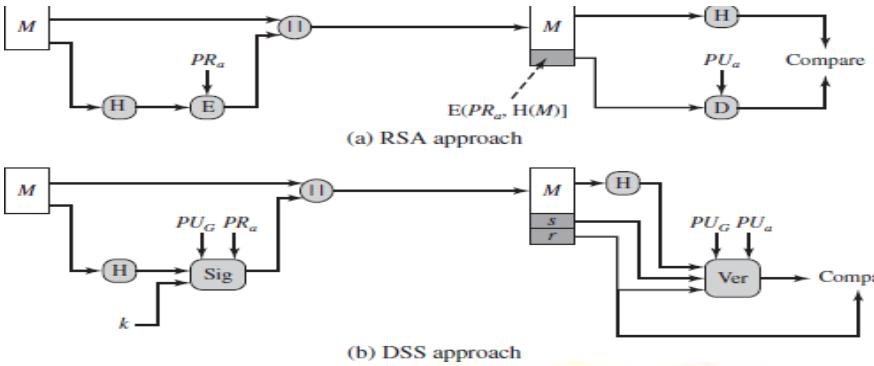
If A is the sender and C is the attacker. Then C finds an efficient signing algorithm that provides anequivalent way of constructing signatures on arbitrary messages.

15. Define Existential forgery

If A is the sender and C is the attacker. Then C forges a signature for at least one message. C has nocontrol over the message. Consequently, this forgery may only be a minor nuisance to A.

16. What are the two approaches of Digital Signature?

- RSA Approach
- DSS Approach



(b)) 10 Marks Questions

1. With the example, explain in detail about Secure Hash Algorithm
2. Explain in detail about HMAC and Digital Signature Standard
3. A) Explain message authentication requirements. What are the attacks related to messagecommunication?
   B) Give a brief note on basic uses of message authentication code.
4. A) Explain the process involved in message digest generation and processing of single block inSHA-512.
   B) What are the approaches of message authentication? Explain them.
5. A) Explain about characteristics of hash functions.
   B) What is the purpose of digital signature? Explain its properties and requirements.
6. A) Write short notes on authentication protocols.

   B) Explain the various types of cryptographic functions with an example.
7. Explain the requirements of digital signatures and also discuss how problems related to digitalsignaturea are taken care by an arbiter?
8. State the need for authentication protocols and explain any three of themDescribe MD5. Compare it with MD4

   Describe SHA-1

   Describe RIPEMD/ HMAC algorithms
9. State and explain the different approaches to message authentication
10. Explain the various methods of producing an authenticaton

UNIT-4

Transport-level Security: Web security considerations, Secure Socket Layer and Transport LayerSecurity, HTTPS, Secure Shell (SSH)
Wireless Network Security: Wireless Security, Mobile Device Security, IEEE802.11 Wireless LAN, IEEE802.11i Wireless LANSecurity

Transport-level Security

1). WEB SECURITY CONSIDERATIONS

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. As such, the security tools and approaches discussed so far in this book are relevant to the issue of Web security. But, as pointed out in [GARF02], the Web presents new challenges not generally appreciated in the context of computer and network security.

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

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

WEB SECURITY THREATS

Table 16.1 provides a summary of the types of security threats faced when using the Web. One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site.

Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server. Issues of server and browser security fall

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

WEB TRAFFIC SECURITY APPROACHES

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

Figure 16.1 illustrates this difference. One way to provide Web security is to use IP security (IPsec) (Figure 16.1a). The advantage of using IPsec is that it is trans- parent to end users and applications and provides a general-purpose solution. Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

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

Table 16.1 A Comparison of Threats on the Web

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

Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS). At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided

as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

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

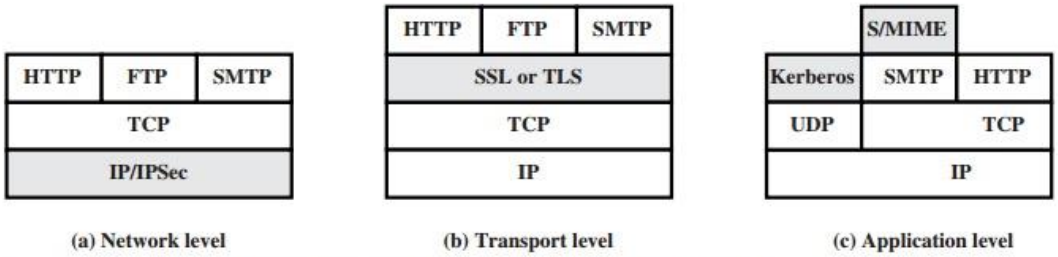


Figure 16.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

2). SECURE SOCKET LAYER (SSL)

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

Secure Socket Layer Protocols:

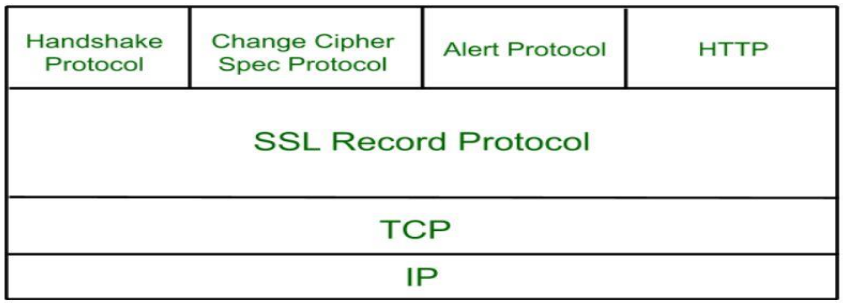
•SSL record protocol

•Handshake protocol

•Change-cipher spec protocol

•Alert protocol

SSL PROTOCOL STACK:



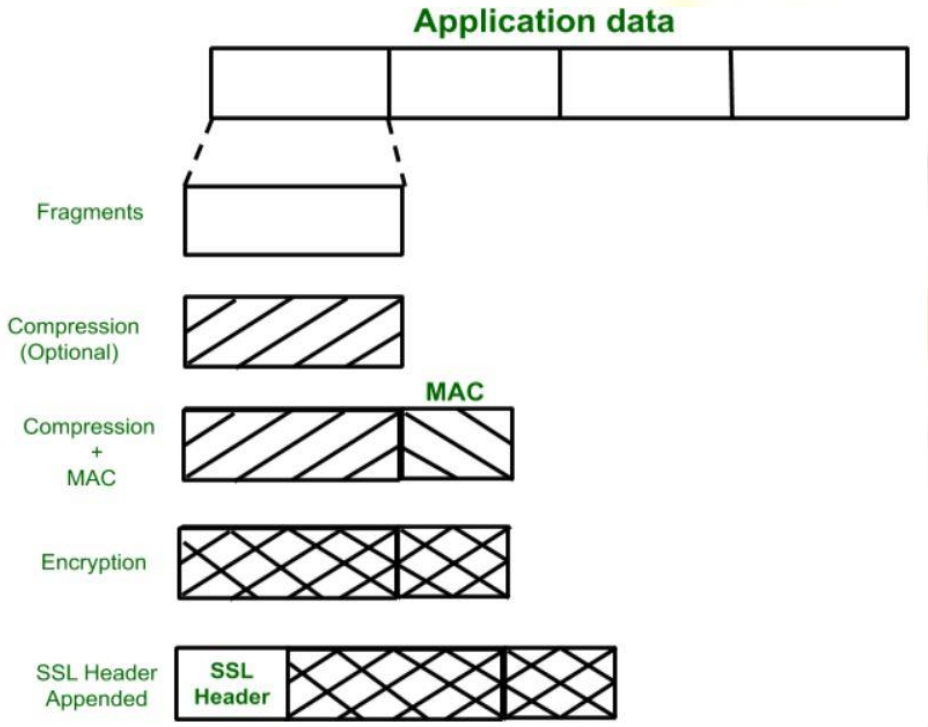
SSL RECORD PROTOCOL:

SSL Record provides two services to SSL connection.

•Confidentiality

•Message Integrity

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



HANDSHAKE PROTOCOL:

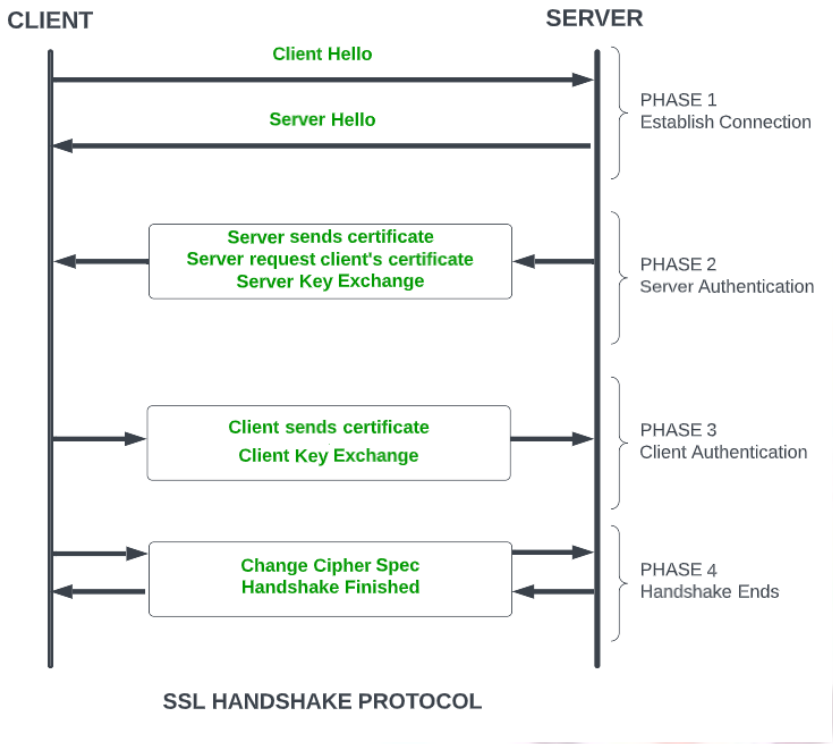
Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

PHASE-1: In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purposes.

PHASE-2: Server sends his certificate and Server-key-exchange. The server end phase-2 by sending the Server-hello-end packet.

PHASE-3: In this phase, Client replies to the server by sending his certificate and Client-exchange-key.

PHASE-4: In Phase-4 Change-cipher suite occurred and after this Handshake Protocol ends.



SSL HANDSHAKE PROTOCOL

CHANGE-CIPHER PROTOCOL:

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

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

ALERT PROTOCOL:

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

The level is further classified into two parts:

WARNING (LEVEL= 1):

This Alert has no impact on the connection between sender and receiver. Some of them are:

BAD CERTIFICATE: When the received certificate is corrupt.

NO CERTIFICATE: When an appropriate certificate is not available.

CERTIFICATE EXPIRED: When a certificate has expired.

CERTIFICATE UNKNOWN: When some other unspecified issue arose in processing the certificate, rendering it unacceptable.

CLOSE NOTIFY: It notifies that the sender will no longer send any messages in the connection.

FATAL ERROR (LEVEL = 2):

This Alert breaks the connection between sender and receiver. The connection will be stopped, cannot be resumed but can be restarted. Some of them are:

HANDSHAKE FAILURE: When the sender is unable to negotiate an acceptable set of security parameters given the options available.

DECOMPRESSION FAILURE: When the decompression function receives improper input.

ILLEGAL PARAMETERS: When a field is out of range or inconsistent with other fields.

BAD RECORD MAC: When an incorrect MAC was received.

UNEXPECTED MESSAGE: When an inappropriate message is received.

The second byte in the Alert protocol describes the error.

SILENT FEATURES OF SECURE SOCKET LAYER:

•The advantage of this approach is that the service can be tailored to the specific needs of the given application.

•Secure Socket Layer was originated by Netscape.

•SSL is designed to make use of TCP to provide reliable end-to-end secure service.

•This is a two-layered protocol.

VERSIONS OF SSL:

SSL 1 – Never released due to high insecurity.

SSL 2 – Released in 1995.

SSL 3 – Released in 1996.

TLS 1.0 – Released in 1999.

TLS 1.1 – Released in 2006.

TLS 1.2 – Released in 2008.

TLS 1.3 – Released in 2018.

3). TRANSPORT LAYER SECURITY:

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

Version Number: The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 1.

Message Authentication Code: Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. For TLS, the MAC calculation encompasses the fields indicated in the following expression:

HMAC hash (MAC write_secret, seq_num || TLSCompressed.type || TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment)

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

Pseudorandom Function: TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The PRF is based on the following data expansion function:

P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) || HMAC_hash(secret, A(2) || seed) || HMAC_hash(secret, A(3) || seed) || ...

where A () is defined as A(0) = seed

A(i) = HMAC_hash (secret, A(i - 1))

The data expansion function makes use of the HMAC algorithm, with either MD5 or SHA- 1 as the underlying hash function. As can be seen, P_hash can be iterated as many times as necessary to

produce the required quantity of data. each iteration involves two executions of HMAC, each of which in turn involves two executions of the underlying hash algorithm.

4). HTTPS

WHAT IS HTTPS?

HTTPS (Hypertext Transfer Protocol Secure) is a secure version of the HTTP protocol that uses the SSL/TLS protocol for encryption and authentication. HTTPS is specified by RFC 2818 (May 2000) and uses port 443 by default instead of HTTP's port 80.

The HTTPS protocol makes it possible for website users to transmit sensitive data such as credit card numbers, banking information, and login credentials securely over the internet. For this reason, HTTPS is especially important for securing online activities such as shopping, banking, and remote work. However, HTTPS is quickly becoming the standard protocol for all websites, whether or not they exchange sensitive data with users.

HOW IS HTTPS DIFFERENT FROM HTTP?

HTTPS adds encryption, authentication, and integrity to the HTTP protocol:

ENCRYPTION: Because HTTP was originally designed as a clear text protocol, it is vulnerable to eavesdropping and man in the middle attacks. By including SSL/TLS encryption, HTTPS prevents data sent over the internet from being intercepted and read by a third party. Through public-key cryptography and the SSL/TLS handshake, an encrypted communication session can be securely set up between two parties who have

never met in person (e.g. a web server and browser) via the creation of a shared secret key.

AUTHENTICATION: Unlike HTTP, HTTPS includes robust authentication via the SSL/TLS protocol. A website's SSL/TLS certificate includes a public key that a web browser can use to confirm that documents sent by the server (such as HTML pages) have been digitally signed by someone in possession of the corresponding private key. If the server's certificate has been signed by a publicly trusted certificate authority (CA), such as SSL.com, the browser will accept that any identifying information included in the certificate has been validated by a trusted third party.

HTTPS websites can also be configured for mutual authentication, in which a web browser presents a client certificate identifying the user. Mutual authentication is useful for situations such as remote work, where it is desirable to include multi-factor authentication, reducing the risk of phishing or other attacks involving credential theft. For more information on configuring client certificates in web browsers, please read this how-to.

INTEGRITY: Each document (such as a web page, image, or JavaScript file) sent to a browser by an HTTPS web server includes a digital signature that a web browser can use to determine that the document has not been altered by a third party or otherwise corrupted while in transit. The server calculates a cryptographic hash of the document's contents, included with its digital certificate, which the browser can independently calculate to prove that the document's integrity is intact.

Taken together, these guarantees of encryption, authentication, and integrity make HTTPS a much safer protocol for browsing and conducting business on the web than HTTP.

What information does HTTPS provide users about website owners?

CAs use three basic validation methods when issuing digital certificates. The validation method used determines the information that will be included in a website's SSL/TLS certificate:

• **DOMAIN VALIDATION (DV)** simply confirms that the domain name covered by the certificate is under the control of the entity that requested the certificate.

• **ORGANIZATION / INDIVIDUAL VALIDATION (OV/IV)** certificates include the validated name of a business or other organization (OV), or an individual person (IV).

• **EXTENDED VALIDATION (EV)** certificates represent the highest standard in internet trust, and require the most effort by the CA to validate. EV certificates are only issued to businesses and other registered organizations, not to individuals, and include the validated name of that organization.

WHY USE HTTPS?

There are multiple good reasons to use HTTPS on your website, and to insist on HTTPS when browsing, shopping, and working on the web as a user:

INTEGRITY AND AUTHENTICATION: Through encryption and authentication, HTTPS protects the integrity of communication between a website and a user's browsers. Your users will know that the data sent from your web server has not been intercepted and/or altered by a third party in transit. And, if you've made the extra investment in EV or OV certificates, they will also be able to tell that the information really came from your business or organization.

PRIVACY: Of course, no one wants intruders scooping up their credit card numbers and passwords while they shop or bank online, and HTTPS is great for preventing that. But would you really want everything else you see and do on the web to be an open book for anyone who feels like snooping (including governments, employers, or someone building a profile to de-anonymize your online activities)? HTTPS plays an important role here too.

USER EXPERIENCE: Recent changes to browser UI have resulted in HTTP sites being flagged as insecure. Do you want your customers' browsers to tell them that your website is "Not Secure" or show them a crossed-out lock when they visit it? Of course not!

COMPATIBILITY: Current browser changes are pushing HTTP ever closer to incompatibility. Mozilla Firefox recently announced an optional HTTPS-only mode, while Google Chrome is steadily

moving to block mixed content (HTTP resources linked to HTTPS pages). When viewed together with browser warnings of "insecurity" for HTTP websites, it's easy to see that the writing is on the wall for HTTP. In 2020, all current major browsers and mobile devices support HTTPS, so you won't lose users by switching from HTTP.

SEO: Search engines (including Google) use HTTPS as a ranking signal when generating search results. Therefore, website owners can get an easy SEO boost just by configuring their web servers to use HTTPS rather than HTTP.

In short, there are no longer any good reasons for public websites to continue to support HTTP. Even the United States government is on board!

HOW DOES HTTPS WORK?

HTTPS adds encryption to the HTTP protocol by wrapping HTTP inside the SSL/TLS protocol (which is why SSL is called a tunneling protocol), so that all messages are encrypted in both directions between two networked computers (e.g. a client and web server). Although an eavesdropper can still potentially access IP addresses, port numbers, domain names, the amount of information exchanged, and the duration of a session, all of the actual data exchanged are securely encrypted by SSL/TLS, including:

• Request URL (which web page was requested by the client)

• Website content

• Query parameters

• Headers

• Cookies

HTTPS also uses the SSL/TLS protocol for authentication. SSL/TLS uses digital documents known as X.509 certificates to bind cryptographic key pairs to the identities of entities such as websites, individuals, and companies. Each key pair includes a private key, which is kept secure, and a public key, which can be widely distributed. Anyone with the public key can use it to:

• Send a message that only the possessor of the private key can decrypt.

• Confirm that a message has been digitally signed by its corresponding private key.

If the certificate presented by an HTTPS website has been signed by a publicly trusted certificate authority (CA), such as SSL.com, users can be assured that the identity of the website has been validated by a trusted and rigorously-audited third party.

WHAT HAPPENS IF MY WEBSITE DOESN'T USE HTTPS?

In 2020, websites that do not use HTTPS or serve mixed content (serving resources like images via HTTP from HTTPS pages) are subject to browser security warnings and errors. Furthermore, these websites unnecessarily compromise their users' privacy and security, and are not preferred by search engine algorithms. Therefore, HTTP and mixed-content websites can expect more browser warnings and errors, lower user trust and poorer SEO than if they had enabled HTTPS.

HOW DO I KNOW IF A WEBSITE USES HTTPS?

An HTTPS URL begins with https:// instead of http://. Modern web browsers also indicate that a user is visiting a secure HTTPS website by displaying a closed padlock symbol to the left of the URL:

In modern browsers like Chrome, Firefox, and Safari, users can click the lock to see if an HTTPS website's digital certificate includes identifying information about its owner.

5). SECURE SHELL (SSH)

WHAT IS SSH?

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.

SSH also refers to the suite of utilities that implement the SSH protocol. Secure Shell provides strong password authentication and public key authentication, as well as encrypted data communications between two computers connecting over an open network, such as the internet.

In addition to providing strong encryption, SSH is widely used by network administrators to manage systems and applications remotely, enabling them to log in to another computer over a network, execute commands and move files from one computer to another.

SSH refers both to the cryptographic network protocol and to the suite of utilities that implement that protocol. SSH uses the client-server model, connecting a Secure Shell client application, which is the end where the session is displayed, with an SSH server, which is the end where the session runs. SSH implementations often include support for application protocols used for terminal emulation or file transfers.

SSH can also be used to create secure tunnels for other application protocols, for example, to securely run X Window System graphical sessions remotely. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

HOW DOES SSH WORK?

Secure Shell was created to replace insecure terminal emulation or login programs, such as Telnet, rlogin (remote login) and rsh (remote shell). SSH enables the same functions -- logging in to and

running terminal sessions on remote systems. SSH also replaces file transfer programs, such as File Transfer Protocol (FTP) and rcp (remote copy).

The most basic use of SSH is to connect to a remote host for a terminal session. The form of that command is the following:

```
ssh UserName@SSHserver.example.com
```

This command will cause the client to attempt to connect to the server named server.example.com, using the user ID UserName. If this is the first time negotiating a connection between the local host and the server, the user will be prompted with the remote host's public key fingerprint and prompted to connect, despite there having been no prior connection:

```
The authenticity of host 'sample.ssh.com' cannot be established.
 DSA key fingerprint is
01:23:45:67:89:ab:cd:ef:ff:fe:dc:ba:98:76:54:32:10.
 Are you sure you want to continue connecting (yes/no)?
```

Answering yes to the prompt will cause the session to continue, and the host key is stored in the local system's known_hosts file. This is a hidden file, stored by default in a hidden directory, called /.ssh/known_hosts, in the user's home directory. Once the host key has been stored in the known_hosts file, the client system can connect directly to that server again without need for any approvals; the host key authenticates the connection.

WHAT IS SSH USED FOR?

Present in all data centers, SSH ships by default with every Unix, Linux and Mac server. SSH connections have been used to secure many different types of communications between a local machine and a remote host, including secure remote access to resources, remote execution of commands, delivery of software patches, and updates and other administrative or management tasks.

In addition to creating a secure channel between local and remote computers, SSH is used to manage routers, server hardware, virtualization platforms, operating systems (OSes), and inside systems management and file transfer applications.

Secure Shell is used to connect to servers, make changes, perform uploads and exit, either using tools or directly through the terminal. SSH keys can be employed to automate access to servers and often are used in scripts, backup systems and configuration management tools.

Designed to be convenient and work across organizational boundaries, SSH keys provide single sign-on (SSO) so that users can move between their accounts without typing a password each time.

While playing pivotal roles in identity management and access management, SSH does more than authenticate over an encrypted connection. All SSH traffic is encrypted. Whether users are transferring a file, browsing the web or running a command, their actions are private.

While it is possible to use SSH with an ordinary user ID and password as credentials, SSH relies more often on public key pairs to authenticate hosts to each other. Individual users must still employ their

user ID and password -- or other authentication methods -- to connect to the remote host itself, but the local machine and the remote machine authenticate separately to each other. This is accomplished by generating a unique public key pair for each host in the communication. A single session requires two public key pairs: one public key pair to authenticate the remote machine to the local machine and a second public key pair to authenticate the local machine to the remote machine.

SECURE SHELL CAPABILITIES:

Functions that SSH enables include the following:

•secure remote access to SSH-enabled network systems or devices for users, as well as automated processes;

•secure and interactive file transfer sessions;

•automated and secured file transfers;

•secure issuance of commands on remote devices or systems; and

•secure management of network infrastructure components.

SSH can be used interactively to enable terminal sessions and should be used instead of the less secure Telnet program. SSH is also commonly used in scripts and other software to enable programs and systems to remotely and securely access data and other resources.

THE HISTORY OF SSH

The first version of SSH appeared in 1995 and was designed by Tatu Ylönen, who was, at the time, a researcher at Helsinki University of Technology and later started SSH Communications Security, a cybersecurity vendor based in Finland.

Over time, various flaws were found in SSH-1. That version is now considered to be deprecated and not safe to use.

SSH-2, the current version of Secure Shell protocols, was adopted as a Standards Track specification by the Internet Engineering Task Force (IETF) in 2006. SSH-2 is not compatible with SSH-1 and uses a Diffie-Hellman key exchange and a stronger integrity check that uses message authentication codes to improve security.

SSH clients and servers can use a number of encryption methods, the mostly widely used being Advanced Encryption Standard (AES) and Blowfish.

There are no known exploitable vulnerabilities in SSH-2, though information leaked by Edward Snowden in 2013 suggested the National Security Agency (NSA) may be able to decrypt some SSH traffic.

SECURE SHELL SECURITY ISSUES

Enterprises using SSH should consider finding ways to manage host keys stored on client systems. These keys can accumulate over time, especially for information technology (IT) staff that needs to be able to access remote hosts for management purposes.

Because the data stored in an SSH known_hosts file can be used to gain authenticated access to remote systems, organizations should be aware of the existence of these files and should have a standard process for retaining control over the files, even after a system is taken out of commission, as the hard drives may have this data stored in plaintext.

Developers should be careful when incorporating SSH commands or functions in a script or other type of program. While it is possible to issue an SSH command that includes a user ID and password to authenticate the user of the local machine to an account on the remote host, doing so may expose the credentials to an attacker with access to the source code.

Shellshock, a security hole in the Bash command processor, can be executed over SSH but is a vulnerability in Bash, not in SSH.

The biggest threat to SSH is poor key management. Without the proper centralized creation, rotation and removal of SSH keys, organizations can lose control over who has access to which resources and when, particularly when SSH is used in automated application-to-application processes.

SSH VS. TELNET

Telnet was one of the first internet application protocols -- the other is FTP. It is used to initiate and maintain a terminal emulation session on a remote host.

SSH and Telnet are functionally similar, with the primary difference being that the SSH protocol uses public key cryptography to authenticate endpoints when setting up a terminal session, as well as for encrypting session commands and output.

While Telnet is primarily used for terminal emulation, SSH can be used to do terminal emulation -- similar to the rlogin command -- as well as for issuing commands remotely as with rsh, transferring files using SSH File Transfer Protocol (SFTP) and tunneling other applications.

SSH VS. SSL/TLS

The Transport Layer Security (TLS) protocol, which updates the Secure Sockets Layer (SSL) protocol, was designed to provide security for network transmissions at the transport layer. The SSH protocol also operates at or just above the transport layer, but there are important differences between the two protocols.

While both rely on public/private key pairs to authenticate hosts, only the server is authenticated with a key pair under TLS. SSH uses a separate key pair to authenticate each connection: one key pair for a connection from a local machine to a remote machine and a second key pair to authenticate the connection from the remote machine to the local machine.

Another difference between SSH and TLS is that TLS enables connections to be encrypted without authentication or authenticated without encryption. SSH encrypts and authenticates all connections.

SSH provides IT and information security (infosec) professionals with a secure mechanism to manage SSH clients remotely. Rather than requiring password authentication to initialize a connection between an SSH client and server, SSH authenticates the devices themselves. This enables IT staff to connect with remote systems and modify SSH configurations, including adding or removing host key pairs in the known_hosts file.

SSH IMPLEMENTATIONS

SSH is an open protocol. It has been implemented for most computing platforms. The open source OpenSSH implementation is the one most commonly found on Linux, Unix and other OSes based on Berkeley Software Distribution (BSD), including Apple's macOS.

OpenSSH was ported to run in Windows PowerShell starting in 2015. In 2018, optional OpenSSH support was added to Windows 10. While SSH is directly accessible by default in most Unix-like OSes, Microsoft's ported version of OpenSSH must be explicitly enabled in the Windows Settings app.

PuTTY is another open source implementation of SSH. While it currently is available for Windows, macOS and Unix/BSD, PuTTY was originally written to run on Windows. It has long been one of the top options for using SSH on a Windows system.

Most implementations of the SSH suite comprise three utilities:

1.slogin (secure login)

2.ssh

3.scp (secure copy)

These are secure versions of the earlier insecure Unix utilities: rlogin, rsh and rcp.

SSH uses public key cryptography to authenticate the remote computer and enables the remote computer to authenticate the user, if necessary.

There are currently dozens of SSH implementations available for various platforms and under a variety of open source and proprietary licenses.

SSH COMMANDS

While there are graphical implementations of SSH, the program is usually invoked at the command line or executed as part of a script. Running the ssh command on its own, with no arguments such as a destination host or user ID, returns a list of SSH command parameters and options.

The most basic form of SSH command is to invoke the program and the destination host name or Internet Protocol (IP) address:

```
ssh server.example.org
```

This will connect to the destination, server.example.org. The destination host will respond by prompting for a password for the user ID of the account under which the client is running. In other

words, if the user ID in use is jsmith, then the remote host will ask for a password associated with the account jsmith on the remote host.

In many cases, the user ID for the remote host will be different, in which case the command should be issued with the remote host user ID, like this:

```
ssh remote_host_userID@server.example.org
```

SSH can also be used from the command line to issue a single command on the remote host and then exit -- for example:

```
ssh example.org ls
```

This command executes the Unix ls command, which lists all contents of the current directory on the remote host. While this example is trivial, it demonstrates that SSH can be used to execute more interesting commands on a remote host. For example, a command can be crafted that initializes a server instance that will give a remote machine access to a single file -- or other resource -- and then terminate the server after the file is accessed by the specified remote host.

In addition to the ssh executable, SSH has other executable commands used at the command line for additional functions, including the following:

•sshd initiates the SSH server, which waits for incoming SSH connection requests and enables authorized systems to connect to the local host.

•ssh-keygen is a program to create a new authentication key pair for SSH, which can be used to automate logins, to implement SSO and to authenticate hosts.

•ssh-copy-id is a program used to copy, install and configure an SSH key on a server to automate passwordless logins and SSO.

•ssh-agent is a helper program that tracks identity keys and their passphrases -- from which SSH derives an encryption key -- and enables the user to use the identity keys to log in to different servers without the need to reenter passwords or passphrases.

•ssh-add is used to add a key to the SSH authentication agent and is used with ssh-agent to implement SSO using SSH.

•scp is a program used for copying files from one computer to another and is an SSH-secured version of rcp.

•sftp is a program used to copy files from one computer to another and is an SSH-secured version of ftp, the original File Transfer Protocol. SFTP has become the preferred mechanism for file sharing over the internet, replacing both FTP and FTP/S (FTP Secure), which is a protocol for using FTP over an SSL/TLS tunnel.

WHAT IS SSH TUNNELING?

SSH tunneling, also known as SSH port forwarding, is a technique that enables a user to open a secure tunnel between a local host and a remote host.

SSH port forwarding redirects network traffic to a particular port/IP address so that a remote host is made directly accessible by applications on the local host. The destination may be on the remote SSH server, or that server may be configured to forward to yet another remote host.

SSH tunnels are powerful tools for IT administrators, as well as malicious actors, because they can transit an enterprise firewall undetected. As a result, there are tools available to prevent unauthorized use of SSH tunnels through a corporate firewall.

6). WIRELESS SECURITY

Like the system's security and data security, keeping a sound knowledge about different wireless security measures is also essential for security professionals. It is because different wireless security mechanisms have a different level of strength and capabilities.

There are automated wireless hacking tools available that have made cybercriminals more powerful. List of some of these tools are:

•AirCrack.

•AirSnort.

•Cain & Able.

•Wireshark.

•NetStumbler etc.

Different hacking techniques include remote accessing, shoulder surfing, wireless router's dashboard accessing, and brute-forcing attack that are used to penetrate wireless security. In this chapter, you will learn about the different security postures that exist in the wireless domain.

WHAT IS WIRELESS SECURITY?

Wireless security revolves around the concept of securing the wireless network from malicious attempts and unauthorized access.

The wireless security can be delivered through different ways such as:

1.Hardware-based: where routers and switches are fabricated with encryption measures protects all wireless communication. So, in this case, even if the data gets compromised by the cybercriminal, they will not be able to decrypt the data or view the traffic's content.

2.Wireless setup of IDS and IPS: helps in detecting, alerting, and preventing wireless networks and sends an alarm to the network administrator in case of any security breach.

3.Wireless security algorithms: such as WEP, WPA, WPA2, and WPA3. These are discussed in the subsequent paragraphs.

WIRED EQUIVALENT PRIVACY (WEP)

Wired Equivalent Privacy (WEP) is the oldest security algorithm of 1999. It uses the initialization vector (IV) method. The first versions of the WEP algorithm were not predominantly strong enough, even when it got released. But the reason for this weak release was because of U.S. limits on exporting different cryptographic technologies, which led the manufacturing companies to restrict their devices to 64-bit encryption only. As the limitation was withdrawn, the 128 bit and 256 bit WEP encryption were developed and came into the wireless security market, though 128 became standard.

WI-FI PROTECTED ACCESS (WPA)

Wi-Fi Protected Access (WPA) was the next Wi-Fi Alliance's project that replaced the WEP standard's increasingly noticeable vulnerabilities. WPA was officially adopted in the year 2003, one year before the retirement of WEP. WPA's most common configuration is with WPA-PSK, which is abbreviated as Pre-Shared Key. WPA uses 256-bit, which was a considerable enhancement above the 64-bit as well as 128-bit keys.

WI-FI PROTECTED ACCESS II (WPA2)

Wi-Fi Protected Access II (WPA2) became official in the year 2006 after WPA got outdated. It uses the AES algorithms as a necessary encryption component as well as uses CCMP (Counter Cipher Mode - Block Chaining Message Authentication Protocol) by replacing TKIP.

WI-FI PROTECTED ACCESS 3 (WPA3)

Wi-Fi Protected Access 3 (WPA3) is the latest and the third iteration of this family developed under Wi-Fi Alliance. It has personal and enterprise security-support features and uses 384-bit Hashed Message Authentication Mode, 256-bit Galois / Counter Mode Protocol (GCMP-256) well as Broadcast/Multicast Integrity Protocol of 256-bit. WPA3 also provides perfect forward secrecy mechanism support.

7). MOBILE DEVICE SECURITY

WHAT IS MOBILE DEVICE SECURITY?

Mobile Device Security refers to the measures designed to protect sensitive information stored on and transmitted by laptops, smartphones, tablets, wearables, and other portable devices. At the root of mobile device security is the goal of keeping unauthorized users from accessing the enterprise network. It is one aspect of a complete enterprise security plan.

WHY IS MOBILE DEVICE SECURITY IMPORTANT?

With more than half of business PCs now mobile, portable devices present distinct challenges to network security, which must account for all of the locations and uses that employees require of the company network. Potential threats to devices include malicious mobile apps, phishing scams, data leakage, spyware, and unsecure Wi-Fi networks. On top of that, enterprises have to account for the possibility of an employee losing a mobile device or the device being stolen. To avoid a security breach, companies should take clear, preventative steps to reduce the risk.

WHAT ARE THE BENEFITS OF MOBILE DEVICE SECURITY?

Mobile device security, or mobile device management, provides the following:

•Regulatory compliance

•Security policy enforcement

•Support of "bring your own device" (BYOD)

•Remote control of device updates

•Application control

•Automated device registration

•Data backup

Above all, mobile device security protects an enterprise from unknown or malicious outsiders being able to access sensitive company data.

HOW DOES MOBILE DEVICE SECURITY WORK?

Securing mobile devices requires a multi-layered approach and investment in enterprise solutions. While there are key elements to mobile device security, each organization needs to find what best fits its network.

To get started, here are some mobile security best practices:

•ESTABILSH, SHARE, AND ENFORCE CLEAR POLICIES AND PROCESSES

Mobile device rules are only as effective as a company's ability to properly communicate those policies to employees. Mobile device security should include clear rules about:

1.What devices can be used

2.Allowed OS levels

3.What the company can and cannot access on a personal phone

4.Whether IT can remote wipe a device

5.Password requirements and frequency for updating passwords

•PASSWORD PROTECTION

One of the most basic ways to prevent unauthorized access to a mobile device is to create a strong password, and yet weak passwords are still a persistent problem that contributes to the majority of data hacks. Another common security problem is workers using the same password for their mobile device, email, and every work-related account. It is critical that employees create strong, unique passwords (of at least eight characters) and create different passwords for different accounts.

•LEVERAGE BIOMETRICS

Instead of relying on traditional methods of mobile access security, such as passwords, some companies are looking to biometrics as a safer alternative. Biometric authentication is when a computer uses measurable biological characteristics, such as face, fingerprint, voice, or iris recognition for identification and access. Multiple biometric authentication methods are now available on smartphones and are easy for workers to set up and use.

•AVOID PUBLIC WI-FI

A mobile device is only as secure as the network through which it transmits data. Companies need to educate employees about the dangers of using public Wi-Fi networks, which are vulnerable to attacks from hackers who can easily breach a device, access the network, and steal data. The best defense is to encourage smart user behavior and prohibit the use of open Wi-Fi networks, no matter the convenience.

•BEWARE OF APPS

Malicious apps are some of the fastest growing threats to mobile devices. When an employee unknowingly downloads one, either for work or personal reasons, it provides unauthorized access to the company's network and data. To combat this rising threat, companies have two options: instruct employees about the dangers of downloading unapproved apps, or ban employees from downloading certain apps on their phones altogether.

•MOBILE DEVICE ENCRYPTION:

Most mobile devices are bundled with a built-in encryption feature. Users need to locate this feature on their device and enter a password to encrypt their device. With this method, data is converted into a code that can only be accessed by authorized users. This is important in case of theft, and it prevents unauthorized access.

WHAT ARE THE DIFFERENT TYPES OF MOBILE DEVICE SECURITY?

There are many aspects to a complete security plan. Common elements of a mobile security solution include the following:

•**ENTERPRISE MOBILE MANAGEMENT PLATFORM:** In addition to setting up internal device policies that protect against unauthorized access, it's equally important to have an Enterprise Mobile Management (EMM) platform that enables IT to gather real-time insights to catch potential threats.

•**EMAIL SECURITY:** Email is the most popular way for hackers to spread ransomware and other malware. To combat such attacks, it's critical for businesses to be armed with advanced email security that can detect, block, and address threats faster; prevent any data loss; and protect important information in transit with end-to-end encryption.

•**ENDPOINT PROTECTION:** This approach protects enterprise networks that are remotely accessed by mobile devices. Endpoint security protects companies by ensuring that portable devices follow security standards and by quickly alerting security teams of detected threats before they can do damage. Endpoint protection also allows IT administrators to monitor operation functions and data backup strategies.

•**VPN:** A virtual private network, or VPN, extends a private network across a public network. This enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network. VPNs' encryption technology allows remote users and branch offices to securely access corporate applications and resources.

•**SECURE WEB GATEWAY:** A secure web gateway protects against online security threats by enforcing company security policies and defending against phishing and malware in real-time. This is especially important for cloud security as this type of protection can identify an attack on one location and immediately stop it at other branches.

•**CLOUD ACCESS SECURITY BROKER:** A cloud access security broker (CASB) is a tool that sits between cloud service consumers and cloud service providers to enforce security, compliance, and governance policies for cloud applications. CASBs help organizations extend the security controls of their on-premises infrastructure to the cloud.

HOW DOES MOBILE DEVICE SECURITY COMPLEMENT EXISTING APPLICATION SECURITY AND NETWORK SECURITY EFFORTS?

In addition to monitoring and protecting against malicious threats to a company's data, mobile device security—when paired with an EMM platform and other network and application security solutions—enables an IT department to remotely manage users and their devices. This capability provides security for all mobile devices connected to a network, while giving IT the option to remotely disable unauthorized users and applications. An EMM also allows IT to remotely wipe company data from a lost or stolen device and to control device updates. All of these measures enhance security significantly.

Making mobile devices secure is not a simple task, but it should be a high priority for any enterprise. To combat the growing threat of cyber-attacks, companies must continually audit their mobile security solutions and consider new security measures as they become available.

8). IEEE 802.11 WIRELESS LAN

Wireless LAN is one of the fastest-growing technologies. IEEE 802.11 refers to the set of standards that define communication for wireless LANs (wireless local area networks, or WLANs). The technology behind 802.11 is branded to consumers as Wi-Fi.

Wireless LAN can be found on college campuses, in office buildings, in hospitals, stock exchanges and in many public areas. It has become popular due to the ease of installation and location freedom with the gaining popularity of laptops.

Wi-Fi is now one of the major forms of communication for many devices, and with home automation increasing, even more, devices are using it. Home Wi-Fi is a big area of usage of technology, with most homes that use broadband connections to the Internet using WiFi access as a key means of communication.

The core of any Wi-Fi system is known as the Access Point, AP. The Wi-Fi access point is essentially the base station that communicates with the Wi-Fi enabled devices - data can then be routed onto a local area network, normally via Ethernet and typically links onto the Internet.

ADVANTAGES

There are various advantages of WLAN, which are as follows −

FAST INSTALLATION AND SIMPLICITY

Installing a wireless LAN system can be fast and easy and can eliminate the need to pull cables through walls, floor, and ceilings.

INCREASED PRODUCTIVITY FOR THE MOBILE EMPLOYEE

The mobile user whose primary computer is a portable computer can change location and always remain connected to the network. This enables the mobile user to travel to various places, let it be meeting rooms, hallways, lobbies, cafeterias, classrooms, and so forth.

REDUCED COST

The initial investment required for wireless LAN hardware is higher than the cost of wired LAN hardware. However, the overall installation expenses and life cycle costs are significantly lower. Long-term cost benefits are greatest in dynamic environments, requiring frequent moves and changes.

MOBILITY AND COLLABORATION

It can stay connected while moving throughout your worksite. Access up-to-the-minute communications and all documents and apps on the network, anywhere, anytime.

ACCESSIBILITY

It can provide network access across your organization, even in areas that have been challenging to reach with the wired network, so your entire team can stay in touch.

EXPANDABILITY

It is used to grow your network efficiently, adding new users and locations without needing to run cables and wires.

GUEST ACCESS

It can offer secure network access to guest users, including customers and business partners while keeping your network resources protected.

9). IEEE 802.11I WIRELESS LAN SECURITY

There are two characteristics of a wired LAN that are not inherent in a wireless LAN.

1. In order to transmit over a wired LAN, a station must be physically connected to the LAN. On the other hand, with a wireless LAN, any station within radio range of the other devices on the LAN can transmit. In a sense, there is a form of authentication with a wired LAN in that it requires some positive and presumably observable action to connect a station to a wired LAN.

2. Similarly, in order to receive a transmission from a station that is part of a wired LAN, the receiving station also must be attached to the wired LAN. On the other hand, with a wireless LAN, any station within radio range can receive. Thus, a wired LAN provides a degree of privacy, limiting reception of data to stations connected to the LAN.

These differences between wired and wireless LANs suggest the increased need for robust security services and mechanisms for wireless LANs. The original specification included a set of security features for privacy and authentication that were quite weak. For privacy, 802.11 defined the Wired Equivalent Privacy (WEP) algorithm. The privacy portion of the 802.11 standard contained major weak- nesses. Subsequent to the development of WEP, the 802.11i task group has developed a set of capabilities to address the WLAN security issues. In order to accelerate the introduction of strong security into WLANs, the Wi-Fi Alliance promulgated Wi-Fi Protected Access (WPA) as a Wi-Fi standard. WPA is a set of security mechanisms that eliminates most 802.11 security issues and was based on the current state of the 802.11i standard. The final form of the 802.11i standard is referred to as Robust Security Network (RSN). The Wi-Fi Alliance certifies vendors in compliance with the full 802.11i specification under the WPA2 program.

IEEE 802.11I SERVICES

The 802.11i RSN security specification defines the following services.

• **AUTHENTICATION:** A protocol is used to define an exchange between a user and an AS that provides mutual authentication and generates temporary keys to be used between the client and the AP over the wireless link.

• **ACCESS CONTROL:** This function enforces the use of the authentication function, routes the messages properly, and facilitates key exchange. It can work with a variety of authentication protocols.

• **PRIVACY WITH MESSAGE INTEGRITY:** MAC-level data (e.g., an LLC PDU) are encrypted along with a message integrity code that ensures that the data have not been altered.

Figure 17.4a indicates the security protocols used to support these services, while Figure 17.4b lists the cryptographic algorithms used for these services.

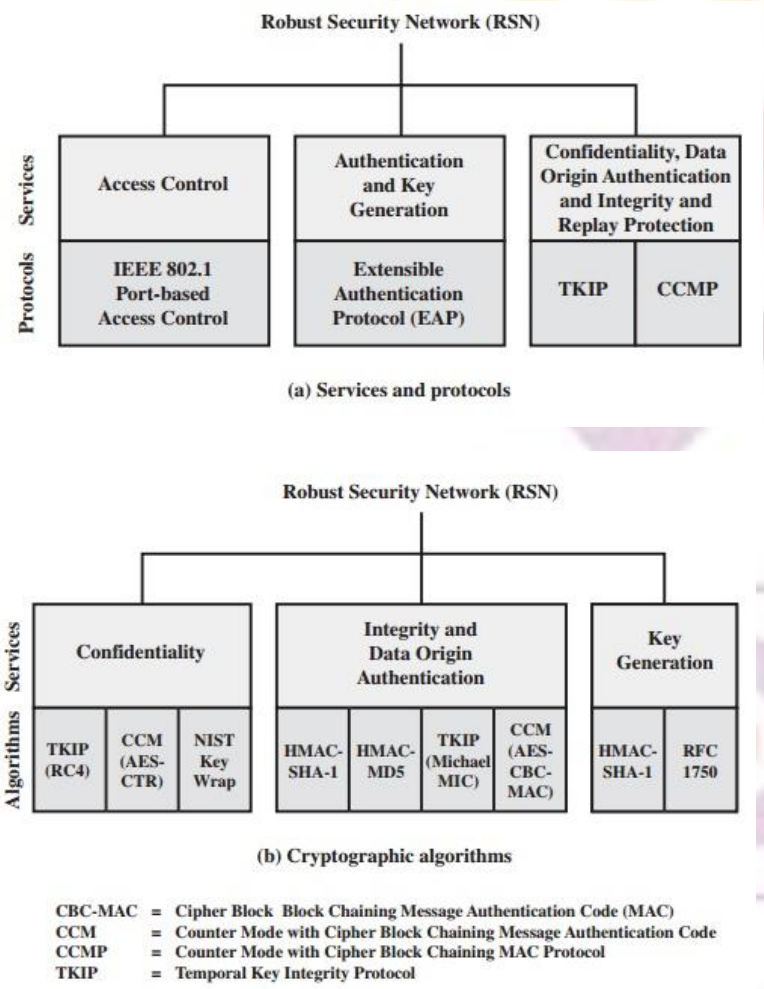


Figure 17.4 Elements of IEEE 802.11i

IEEE 802.11I PHASEES OF OPERATION

The operation of an IEEE 802.11i RSN can be broken down into five distinct phases of operation. The exact nature of the phases will depend on the configuration and the end points of the communication. Possibilities include (see Figure 17.3):

1. Two wireless stations in the same BSS communicating via the access point (AP) for that BSS.

2. Two wireless stations (STAs) in the same ad hoc IBSS communicating directly with each other.

3. Two wireless stations in different BSSs communicating via their respective APs across a distribution system.

4. A wireless station communicating with an end station on a wired network via its AP and the distribution system.

IEEE 802.11i security is concerned only with secure communication between the STA and its AP. In case 1 in the preceding list, secure communication is assured if each STA establishes secure communications with the AP. Case 2 is similar, with the AP functionality residing in the STA. For case 3, security is not provided across the distribution system at the level of IEEE 802.11, but only within each BSS. End- to-end security (if required) must be provided at a higher layer. Similarly, in case 4, security is only provided between the STA and its AP.

With these considerations in mind, Figure 17.5 depicts the five phases of oper- ation for an RSN and maps them to the network components involved. One new component is the authentication server (AS). The rectangles indicate the exchange of sequences of MPDUs. The five phases are defined as follows.

• **DISCOVERY:** An AP uses messages called Beacons and Probe Responses to advertise its IEEE 802.11i security policy. The STA uses these to identify an AP for a WLAN with which it wishes to communicate. The STA associates with the AP, which it uses to select the cipher suite and authentication mecha- nism when the Beacons and Probe Responses present a choice.

AUTHENTICATION: During this phase, the STA and AS prove their identities to each other. The AP blocks non-authentication traffic between the STA and AS until the authentication transaction is successful. The AP does not participate in the authentication transaction other than forwarding traffic between the STA and AS.

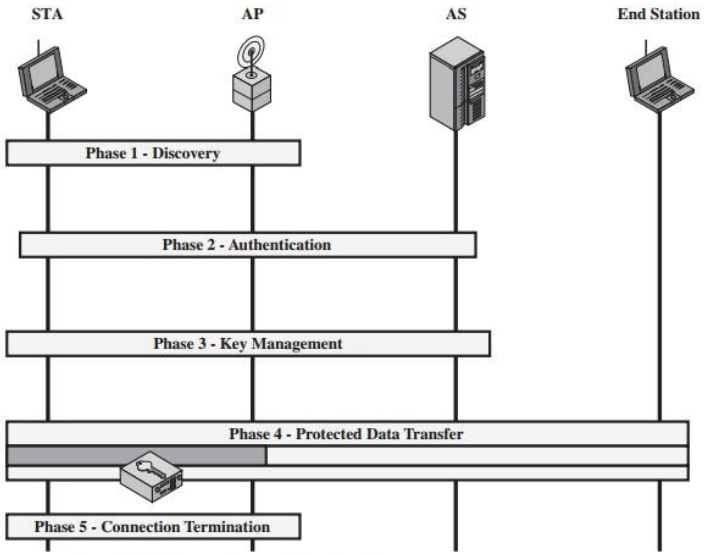


Figure 17.5 IEEE 802.11i Phases of Operation

• **KEY GENERATION AND DISTRIBUTION:** The AP and the STA perform several operations that cause cryptographic keys to be generated and placed on the AP and the STA. Frames are exchanged between the AP and STA only.

• **PROTECTED DATA TRANSFER:** Frames are exchanged between the STA and the end station through the AP. As denoted by the shading and the encryption module icon, secure data transfer occurs between the STA and the AP only; security is not provided end-to-end.

• **CONNECTION TERMINATION:** The AP and STA exchange frames. During this phase, the secure connection is torn down and the connection is restored to the original state.

DISCOVERY PHASE

We now look in more detail at the RSN phases of operation, beginning with the discovery phase, which is illustrated in the upper portion of Figure 17.6. The purpose of this phase is for an STA and an AP to recognize each other, agree on a set of secu- rity capabilities, and establish an association for future communication using those security capabilities.

SECURITY CAPABILITIES During this phase, the STA and AP decide on specific techniques in the following areas:

• Confidentiality and MPDU integrity protocols for protecting unicast traffic (traffic only between this STA and AP)

• Authentication method

• Cryptography key management approach

Confidentiality and integrity protocols for protecting multicast/broadcast traf- fic are dictated by the AP, since all STAs in a multicast group must use the same pro- tocols and ciphers. The specification of

a protocol, along with the chosen key length (if variable) is known as a cipher suite. The options for the confidentiality and integrity cipher suite are

• WEP, with either a 40-bit or 104-bit key, which allows backward compatibility with older IEEE 802.11 implementations

• TKIP

• CCMP

• Vendor-specific methods

The other negotiable suite is the authentication and key management (AKM) suite, which defines (1) the means by which the AP and STA perform mutual authentication and (2) the means for deriving a root key from which other keys may be generated. The possible AKM suites are

• IEEE 802.1X

• Pre-shared key (no explicit authentication takes place and mutual authentica- tion is implied if the STA and AP share a unique secret key)

• Vendor-specific methods

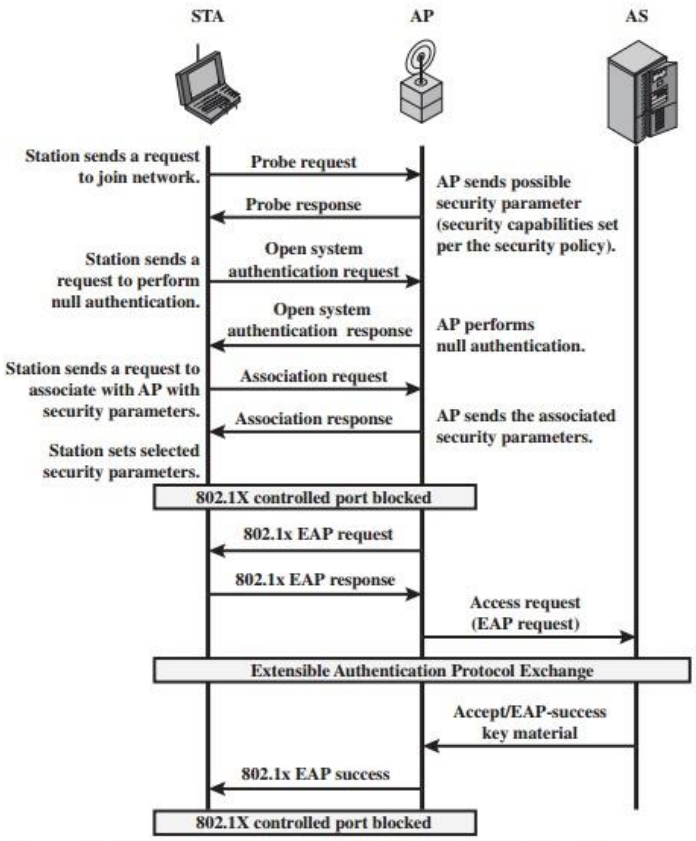


Figure 17.6 IEEE 802.11i Phases of Operation: Capability Discovery, Authentication, and Association

MPDU EXCHANGE The discovery phase consists of three exchanges.

• **NETWORK AND SECURITY CAPABILITY DISCOVERY:** During this exchange, STAs discover the existence of a network with which to communicate. The AP either periodically broadcasts its security capabilities (not shown in figure), indicated by RSN IE (Robust Security Network Information Element), in a specific channel through the Beacon frame; or responds to a station's Probe Request through a Probe Response frame. A wireless station may discover available access points and corresponding security capabilities by either passively monitoring the Beacon frames or actively probing every channel.

OPEN SECURITY AUTHENTICATION: The purpose of this frame sequence, which provides no security, is simply to maintain backward compatibility with the IEEE 802.11 state machine, as implemented in existing IEEE 802.11 hard- ware. In essence, the two devices (STA and AP) simply exchange identifiers.

ASSOCIATION: The purpose of this stage is to agree on a set of security capabilities to be used. The STA then sends an Association Request frame to the AP. In this frame, the STA specifies one set of matching capabilities (one authentication and key management suite, one pairwise cipher suite, and one group- key cipher suite) from among those advertised by the AP. If there is no match in

capabilities between the AP and the STA, the AP refuses the Association Request. The STA blocks it too, in case it has associated with a rogue AP or someone is inserting frames illicitly on its channel. As shown in Figure 17.6, the IEEE 802.1X controlled ports are blocked, and no user traffic goes beyond the AP. The concept of blocked ports is explained subsequently.

AUTHENTICATION PHASE

As was mentioned, the authentication phase enables mutual authentication between an STA and an authentication server (AS) located in the DS. Authentication is designed to allow only authorized stations to use the network and to provide the STA with assurance that it is communicating with a legitimate network.

IEEE 802.1X ACCESS CONTROL APPROACH IEEE 802.11i makes use of another standard that was designed to provide access control functions for LANs. The standard is IEEE 802.1X, Port-Based Network Access Control. The authentication protocol that is used, the Extensible Authentication Protocol (EAP), is defined in the IEEE 802.1X standard. IEEE 802.1X uses the terms supplicant, authenticator, and authentication server (AS). In the context of an 802.11 WLAN, the first two terms correspond to the wireless station and the AP. The AS is typically a separate device on the wired side of the network (i.e., accessible over the DS) but could also reside directly on the authenticator.

Before a supplicant is authenticated by the AS using an authentication proto- col, the authenticator only passes control or authentication messages between the supplicant and the AS; the 802.1X control channel is unblocked, but the 802.11 data channel is blocked. Once a supplicant is authenticated and keys are provided, the authenticator can forward data from the supplicant, subject to predefined access control limitations for the supplicant to the network. Under these circumstances, the data channel is unblocked.

As indicated in Figure 17.7, 802.1X uses the concepts of controlled and uncontrolled ports. Ports are logical entities defined within the authenticator and refer to physical network connections. For a WLAN, the authenticator (the AP) may have only two physical ports: one connecting to the DS and one for wireless communication within its BSS. Each logical port is mapped to one of these two physical ports. An uncontrolled port allows the exchange of PDUs between the supplicant and the other AS, regardless of the authentication state of the supplicant. A controlled port allows the exchange of PDUs between a supplicant and other systems on the LAN only if the current state of the supplicant authorizes such an exchange.

• The 802.1X framework, with an upper-layer authentication protocol, fits nicely with a BSS architecture that includes a number of wireless stations and an AP.

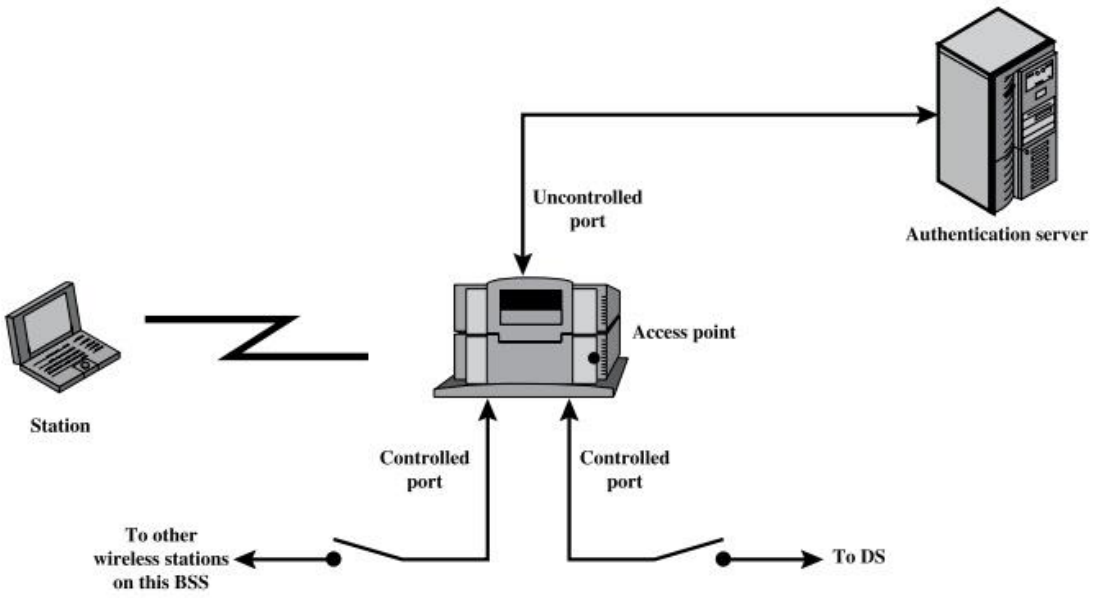


Figure 17.7 802.1X Access Control

However, for an IBSS, there is no AP. For an IBSS, 802.11i provides a more complex solution that, in essence, involves pairwise authentication between stations on the IBSS.

MPDU EXCHANGE The lower part of Figure 17.6 shows the MPDU exchange dictated by IEEE 802.11 for the authentication phase. We can think of authentication phase as consisting of the following three phases.

• **CONNECT TO AS:** The STA sends a request to its AP (the one with which it has an association) for connection to the AS. The AP acknowledges this request and sends an access request to the AS.

• **EAP EXCHANGE:** This exchange authenticates the STA and AS to each other. A number of alternative exchanges are possible, as explained subsequently.

• **SECURE KEY DELVERY:** Once authentication is established, the AS generates a master session key (MSK), also known as the Authentication, Authorization, and Accounting (AAA) key and sends it to the STA. As explained subsequently, all the cryptographic keys needed by the STA for secure communication with its AP are generated from this MSK. IEEE 802.11i does not prescribe a method for secure delivery of the MSK but relies on EAP for this. Whatever method is used, it involves the transmis- sion of an MPDU containing an encrypted MSK from the AS, via the AP, to the AS.

EAP EXCHANGE As mentioned, there are a number of possible EAP exchanges that can be used during the authentication phase. Typically, the message flow between STA and AP employs the EAP over LAN (EAPOL) protocol, and the message flow between the AP and AS uses the Remote Authentication Dial In User Service (RADIUS) protocol, although other options are available for both STA-to- AP and AP-to-AS exchanges. [FRAN07] provides the following summary of the authentication exchange using EAPOL and RADIUS.

1. The EAP exchange begins with the AP issuing an EAP-Request/Identity frame to the STA.

2. The STA replies with an EAP-Response/Identity frame, which the AP receives over the uncontrolled port. The packet is then encapsulated in RADIUS over EAP and passed on to the RADIUS server as a RADIUS-Access-Request packet.

3. The AAA server replies with a RADIUS-Access-Challenge packet, which is passed on to the STA as an EAP-Request. This request is of the appropriate authentication type and contains relevant challenge information.

4. The STA formulates an EAP-Response message and sends it to the AS. The response is translated by the AP into a Radius-Access-Request with the response to the challenge as a data field. Steps 3 and 4 may be repeated multi- ple times, depending on the EAP method in use. For TLS tunneling methods, it is common for authentication to require 10 to 20 round trips.

5. The AAA server grants access with a Radius-Access-Accept packet. The AP issues an EAP-Success frame. (Some protocols require confirmation of the EAP success inside the TLS tunnel for authenticity validation.) The controlled port is authorized, and the user may begin to access the network.

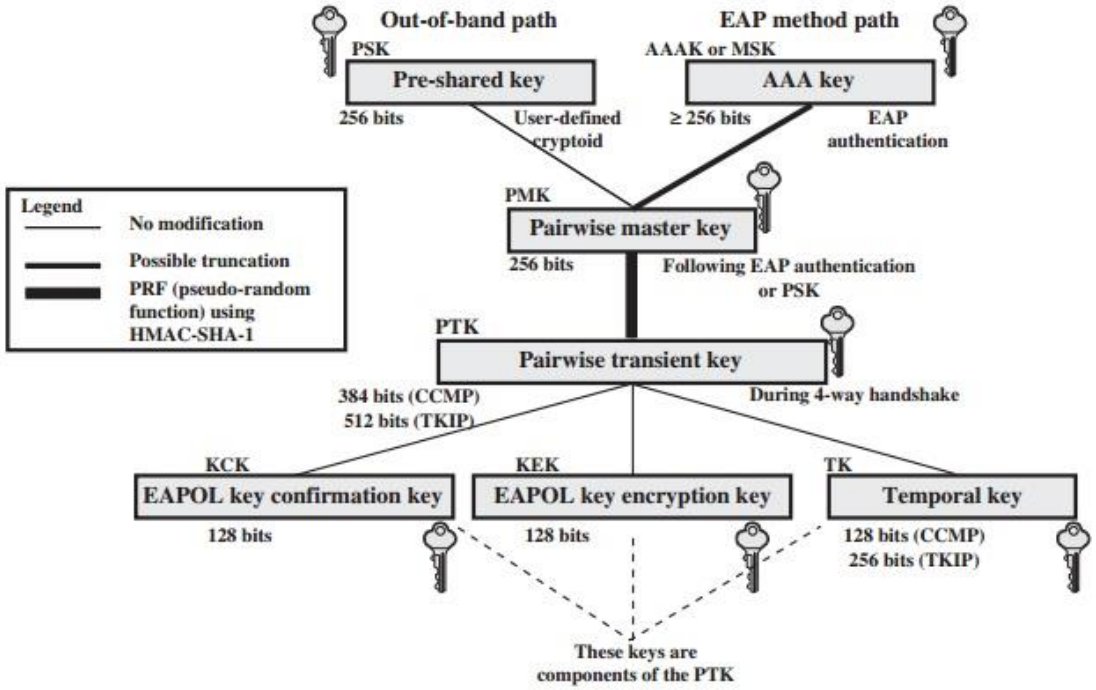
Note from Figure 17.6 that the AP controlled port is still blocked to general user traffic. Although the authentication is successful, the ports remain blocked until the temporal keys are installed in the STA and AP, which occurs during the 4-Way Handshake.

KEY MANAGEMENT PHASE

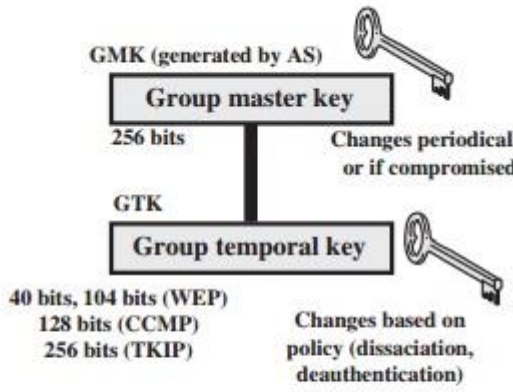
During the key management phase, a variety of cryptographic keys are generated and distributed to STAs. There are two types of keys: pairwise keys used for communication between an STA and an AP and group keys used for multicast communication. Figure 17.8, based on [FRAN07], shows the two key hierarchies, and Table 17.3 defines the individual keys.

PAIRWISE KEYS Pairwise keys are used for communication between a pair of devices, typically between an STA and an AP. These keys form a hierarchy beginning with a master key from which other keys are derived dynamically and used for a limited period of time.

At the top level of the hierarchy are two possibilities. A pre-shared key (PSK) is a secret key shared by the AP and a STA and installed in some fashion outside the scope of IEEE 802.11i. The other alternative is the master session key (MSK), also known as the AAAK, which is generated using the IEEE 802.1X protocol dur- ing the authentication phase, as described previously. The actual method of key generation depends on the details of the authentication protocol used. In either



(a) Pairwise key hierarchy



(b) Group key hierarchy

Figure 17.8 IEEE 802.11i Key Hierarchies

case (PSK or MSK), there is a unique key shared by the AP with each STA with which it communicates. All the other keys derived from this master key are also unique between an AP and an STA. Thus, each STA, at any time, has one set of keys, as depicted in the hierarchy of Figure 17.8a, while the AP has one set of such keys for each of its STAs.

The pairwise master key (PMK) is derived from the master key. If a PSK is used, then the PSK is used as the PMK; if a MSK is used, then the PMK is derived from the MSK by truncation (if necessary). By

the end of the authentication phase, marked by the 802.1x EAP Success message (Figure 17.6), both the AP and the STA have a copy of their shared PMK.

Table 17.3 IEEE 802.11i Keys for Data Confidentiality and Integrity Protocols

Abbrev-iation	Name	Description / Purpose	Size (bits)	Type
AAA Key	Authentication, Accounting, and Authorization Key	Used to derive the PMK. Used with the IEEE 802.1X authentication and key management approach. Same as MMSK.	≥ 256	Key generation key, root key
PSK	Pre-shared Key	Becomes the PMK in pre-shared key environments.	256	Key generation key, root key
PMK	Pairwise Master Key	Used with other inputs to derive the PTK.	256	Key generation key
GMK	Group Master Key	Used with other inputs to derive the GTK.	128	Key generation key
PTK	Pair-wise Transient Key	Derived from the PMK. Comprises the EAPOL-KCK, EAPOL-KEK, and TK and (for TKIP) the MIC key.	512 (TKIP) 384 (CCMP)	Composite key
TK	Temporal Key	Used with TKIP or CCMP to provide confidentiality and integrity protection for unicast user traffic.	256 (TKIP) 128 (CCMP)	Traffic key
GTK	Group Temporal Key	Derived from the GMK. Used to provide confidentiality and integrity protection for multicast/broadcast user traffic.	256 (TKIP) 128 (CCMP) 40, 104 (WEP)	Traffic key
MIC Key	Message Integrity Code Key	Used by TKIP's Michael MIC to provide integrity protection of messages.	64	Message integrity key
EAPOL-KCK	EAPOL-Key Confirmation Key	Used to provide integrity protection for key material distributed during the 4-Way Handshake.	128	Message integrity key
EAPOL-KEK	EAPOL-Key Encryption Key	Used to ensure the confidentiality of the GTK and other key material in the 4-Way Handshake.	128	Traffic key / key encryption key
WEP Key	Wired Equivalent Privacy Key	Used with WEP.	40, 104	Traffic key

The PMK is used to generate the pairwise transient key (PTK), which in fact consists of three keys to be used for communication between an STA and AP after they have been mutually authenticated. To derive the PTK, the HMAC-SHA-1 function is applied to the PMK, the MAC addresses of the STA and AP, and nonces generated when needed. Using the STA and AP addresses in the generation of the PTK provides protection against session hijacking and impersonation; using nonces provides additional random keying material.

The three parts of the PTK are as follows.

• **EAP OVER LAN (EAPOL) KEY CONFIRAMATION KEY (EAPOL-KCK):** Supports the integrity and data origin authenticity of STA-to-AP control frames during operational setup of an RSN. It also performs an access control function: proof-of-possession of the PMK. An entity that possesses the PMK is autho- rized to use the link.

• **EAPOL KEY ENCRYPTION KEY (EAPOL-KEK):** Protects the confidentiality of keys and other data during some RSN association procedures.

• **TEMPORAL KEY (TK):** Provides the actual protection for user traffic.

GROUP KEYS Group keys are used for multicast communication in which one STA sends MPDU's to multiple STAs. At the top level of the group key hierarchy is the group master key (GMK). The GMK is a key-generating key used with other inputs to derive the group temporal key (GTK). Unlike the PTK, which is generated using material from both AP and STA, the GTK is generated by the AP and transmitted to its associated STAs. Exactly how this GTK is generated is undefined. IEEE 802.11i, however, requires that its value is computationally indistinguishable from random. The GTK is distributed securely using the pairwise keys that are already established. The GTK is changed every time a device leaves the network.

PAIRWISE KEY DISTRIBUTION The upper part of Figure 17.9 shows the MPDU exchange for distributing pairwise keys. This exchange is known as the 4-way handshake. The STA and SP use this handshake to confirm the existence of the PMK, verify the selection of the cipher suite, and derive a fresh PTK for the following data session. The four parts of the exchange are as follows.

• **AP: STA:** Message includes the MAC address of the AP and a nonce (Anonce)

• **STA: AP:** The STA generates its own nonce (Snonce) and uses both nonces and both MAC addresses, plus the PMK, to generate a PTK. The STA then sends a message containing its MAC address and Snonce, enabling the AP to generate the same PTK. This message includes a message integrity code (MIC)2 using HMAC-MD5 or HMAC-SHA-1-128. The key used with the MIC is KCK.

• **AP: STA:** The AP is now able to generate the PTK. The AP then sends a message to the STA, containing the same information as in the first message, but this time including a MIC.

• **STA: AP:** This is merely an acknowledgment message, again protected by a MIC.

GROUP KEY DISTRIBUTION For group key distribution, the AP generates a GTK and distributes it to each STA in a multicast group. The two-message exchange with each STA consists of the following:

AP: STA: This message includes the GTK, encrypted either with RC4 or with AES. The key used for encryption is KEK. A MIC value is appended.

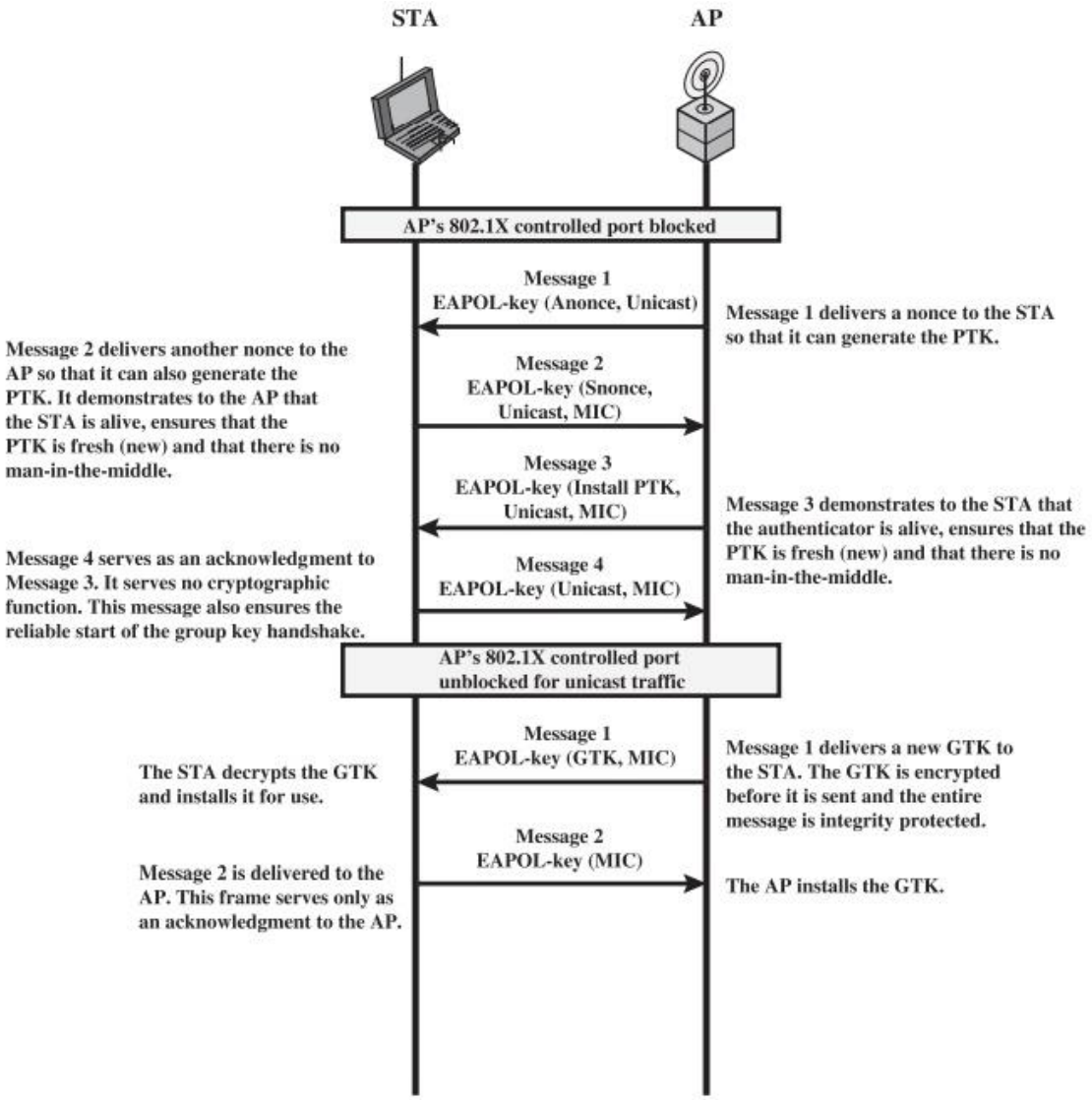


Figure 17.9 IEEE 802.11i Phases of Operation: Four-Way Handshake and Group Key Handshake

• **STA: AP:** The STA acknowledges receipt of the GTK. This message includes a MIC value.

PROTECTED DATA TRANSFER PHASE

IEEE 802.11i defines two schemes for protecting data transmitted in 802.11 MPDUs: the Temporal Key Integrity Protocol (TKIP), and the Counter Mode-CBC MAC Protocol (CCMP).

TKIP TKIP is designed to require only software changes to devices that are implemented with the older wireless LAN security approach called Wired Equivalent Privacy (WEP). TKIP provides two services:

• **MESSAGE INTEGRITY:** TKIP adds a message integrity code (MIC) to the 802.11 MAC frame after the data field. The MIC is generated by an algorithm, called Michael, that computes a 64-bit value using as input the source and destination MAC address values and the Data field, plus key material.

• **DATA CONFIDENTIALITY:** Data confidentiality is provided by encrypting the MPDU plus MIC value using RC4.

The 256-bit TK (Figure 17.8) is employed as follows. Two 64-bit keys are used with the Michael message digest algorithm to produce a message integrity code. One key is used to protect STA-to-AP messages, and the other key is used to protect AP-to-STA messages. The remaining 128 bits are truncated to generate the RC4 key used to encrypt the transmitted data.

For additional protection, a monotonically increasing TKIP sequence counter (TSC) is assigned to each frame. The TSC serves two purposes. First, the TSC is included with each MPDU and is protected by the MIC to protect against replay attacks. Second, the TSC is combined with the session TK to produce a dynamic encryption key that changes with each transmitted MPDU, thus making cryptanalysis more difficult.

CCMP CCMP is intended for newer IEEE 802.11 devices that are equipped with the hardware to support this scheme. As with TKIP, CCMP provides two services:

•**MESSAGE INTEGRITY:** CCMP uses the cipher-block-chaining message authentication code (CBC-MAC), described in Chapter 12.

• **DATA CONFIDENTIALITY:** CCMP uses the CTR block cipher mode of operation with AES for encryption. CTR is described in Chapter 6.

The same 128-bit AES key is used for both integrity and confidentiality. The scheme uses a 48-bit packet number to construct a nonce to prevent replay attacks.

THE IEEE 802.11I PSEUDORANDOM FUNCTION

At a number of places in the IEEE 802.11i scheme, a pseudorandom function (PRF) is used. For example, it is used to generate nonces, to expand pairwise keys, and to generate the GTK. Best security practice dictates that different pseudorandom number streams be used for these different purposes. However, for implementation efficiency, we would like to rely on a single pseudorandom number generator function.

The PRF is built on the use of HMAC-SHA-1 to generate a pseudorandom bit stream. Recall that HMAC-SHA-1 takes a message (block of data) and a key of length at least 160 bits and produces a 160-bit hash value. SHA-1 has the property that the change of a single bit of the input produces a new

hash value with no appar- ent connection to the preceding hash value. This property is the basis for pseudorandom number generation.

The IEEE 802.11i PRF takes four parameters as input and produces the desired number of random bits. The function is of the form PRF(K, A, B, Len), where

K = a secret key

A = a text string specific to the application (e.g., nonce generation or pair- wise key expansion)

B = some data specific to each case

Len = desired number of pseudorandom bits

For example, for the pairwise transient key for CCMP:

PTK = PRF(PMK, "Pairwise key expansion", min(AP– Addr, STA–Addr)|| max(AP–Addr, STA–Addr) | min (Anonce, Snonce) | max(Anonce, Snonce), 384)

So, in this case, the parameters are

K = PMK

A = the text string "Pairwise key expansion"

B = a sequence of bytes formed by concatenating the two MAC addresses and the two nonces

Len = 384 bits

Similarly, a nonce is generated by

Nonce = PRF (Random Number, "Init Counter", MAC | Time, 256)

where Time is a measure of the network time known to the nonce generator.

The group temporal key is generated by

GTK = PRF (GMK, "Group key expansion", MAC | Gnonce, 256

Figure 17.10 illustrates the function PRF(K, A, B, Len). The parameter K serves as the key input to HMAC. The message input consists of four items concatenated together: the parameter A, a byte with value 0, the parameter B, and a counter i. The counter is initialized to 0. The HMAC algorithm is run once, producing a 160-bit hash value. If more bits are required, HMAC is run again with the same inputs, except that i is incremented each time until the necessary number of bits is generated. We can express the logic as

PRF(K, A, B, Len)

R ‹- null string

for i ‹- 0 to ((Len + 159)/160 – 1) do

R ‹- R | HMAC–SHA–1(K, A ||0 |B |i)

Return Truncate–to–Len(R, Len)

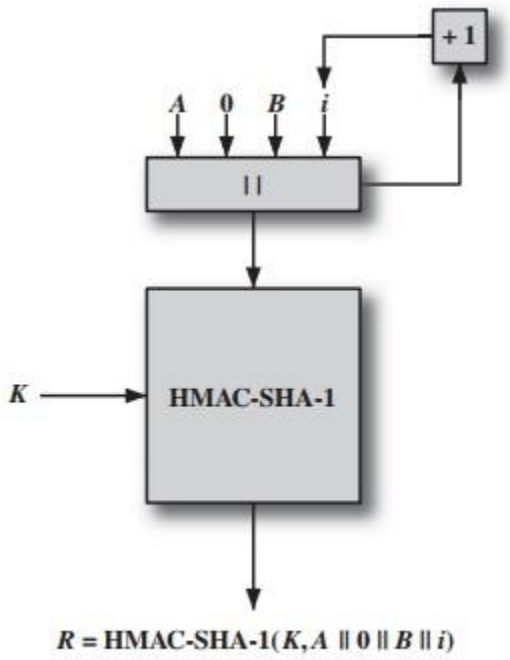


Figure 17.10 IEEE 802.11i Pseudorandom Function

UNIT-5

E-Mail Security: Pretty Good Privacy, S/MIME **IPSecurity:** IP Security overview, IP Security architecture, Authentication Header, Encapsulating security payload, Combining security associations, Internet Key Exchange

Case Studies on Cryptography and security: Secure Multiparty Calculation, Virtual Elections, Single sign On, Secure Inter-branch Payment Transactions, Cross site Scripting Vulnerability.

E-Mail Security

1). PRETTY GOOD PRIVACY

In virtually all distributed environments, electronic mail is the most heavily used network-based application. But current email services are roughly like "postcards", anyone who wants could pick it up and have a look as it's in transit or sitting in the recipients mailbox. PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. With the explosively growing reliance on electronic mail for every conceivable purpose, there grows a demand for authentication and confidentiality services. The Pretty Good Privacy (PGP) secure email program, is a remarkable phenomenon, has grown explosively and is now widely used. Largely the effort of a single person, Phil Zimmermann, who selected the best available crypto algorithms to use & integrated them into a single program, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. It is independent of government organizations and runs on a wide range of systems, in both free & commercial versions. There are **five** important services in PGP

Authentication(Sign/Verify)

Confidentiality(Encryption/Decryption)

Compression

Email Compatibility

Segmentation and Reassembly

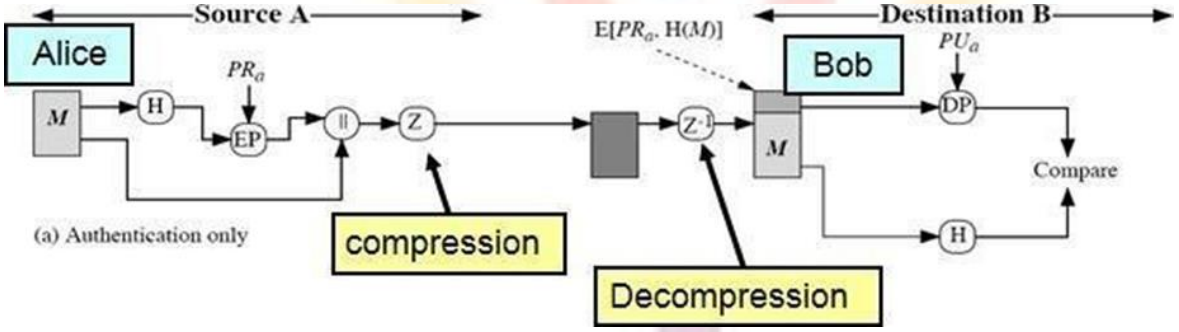
The last three are **transparent to** the user

PGP Notations:

Ks	=session key used in symmetric encryption scheme
PRa	=private key of user A, used in public-key encryption scheme

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

PGP Operation-Authentication



1.sender creates message

2.use SHA-1 to generate 160-bit hash of message

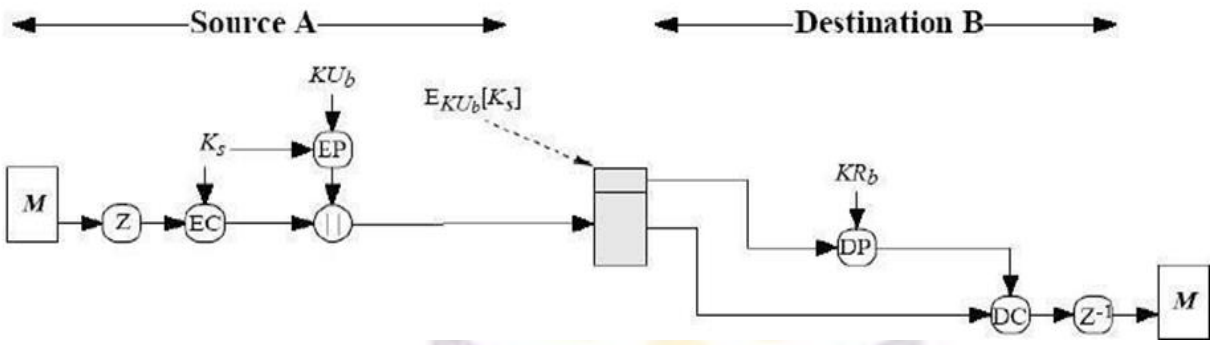
3.signed hash with RSA using sender's private key, and is attached to message

4.receiver uses RSA with sender's public key to decrypt and recover hash code

5.receiver verifies received message using hash of it and compares with decrypted hash code

PGP Operation-Confidentiality

PGP Operation- Confidentiality



Sender:

1.Generates message and a random number (session key) only for this message

2.Encrypts message with the session key using AES, 3DES, IDEA or CAST-128

3.Encrypts session key itself with recipient's public key using RSA

4.Attaches it to message

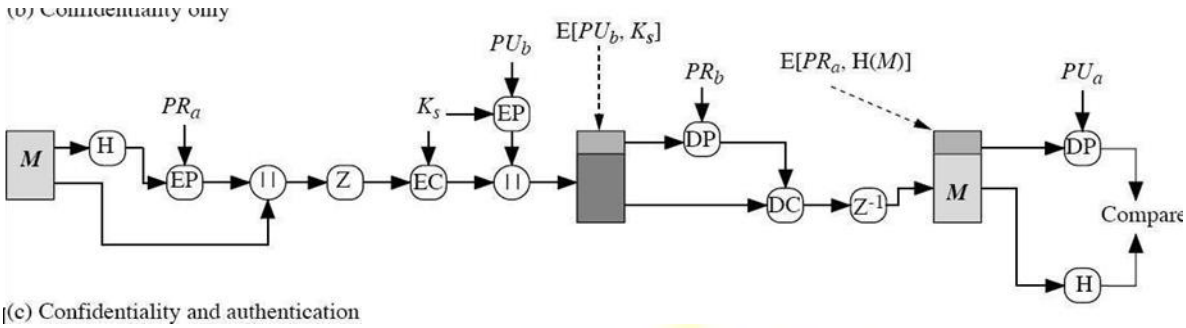
Receiver:

1.Recovers session key by decrypting using his private key

2.Decrypts message using the session key

Confidentiality service provides no assurance to the receiver as to the identity ofsender (i.e. no authentication). Only provides confidentiality for sender that only therecipient can read the message (and no one else) can use both services on same message o create signature & attach to message o encrypt both message & signature o attach RSA/ElGamal encrypted session key
o *is called* ***authenticated confidentiality*****.**

PGP Operation – Confidentiality & Authentication



PGP Operation-Compression

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

The signature is generated before compression for two reasons:

1.so that one can store only the uncompressed message together with signature for later verification

2.Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm as the PGP compression algorithm is not deterministic

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

PGP Operation-Email Compatibility

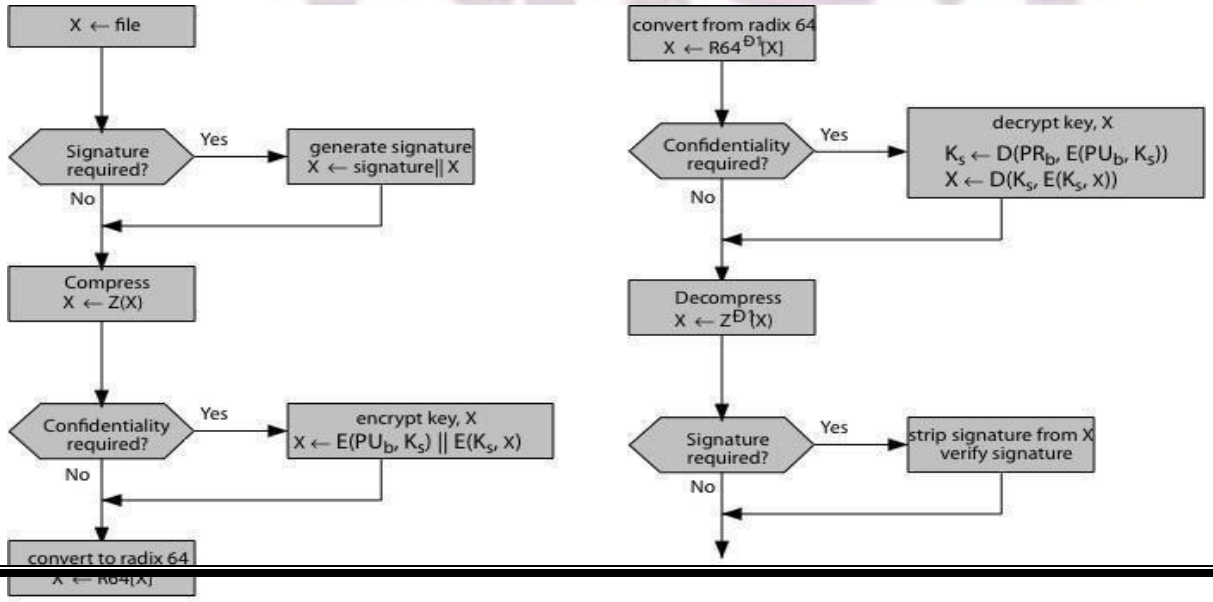
When PGP is used, at least part of the block to be transmitted is encrypted, and thus consists of a stream of arbitrary 8-bit octets. However many electronic mail systems only permit the use of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. It uses radix-64 conversion, in which each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors. The use of radix 64 expands a message by 33%, but still an overall compression of about one-third can be achieved.

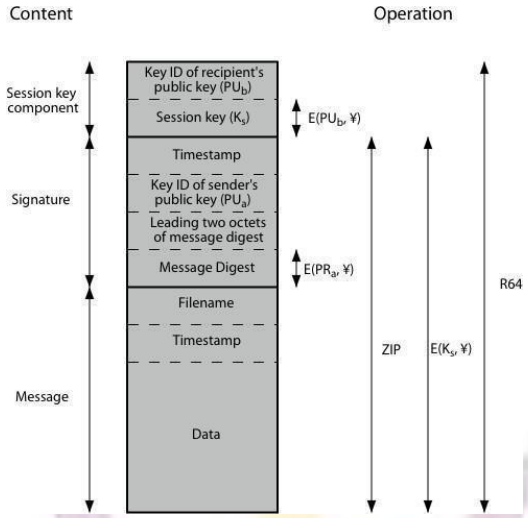
PGP Operation-Segmentation/Reassembly

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

PGP Operations-Summary

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





PGP Message Format

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

1.Timestamp: The time at which the signature was made.
2.Message digest: The 160-bit SHA-1 digest, encrypted with the sender's private signature key.
3. Leading two octets of message digest: To enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication, by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.
4. Key ID of sender's public key: Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest.

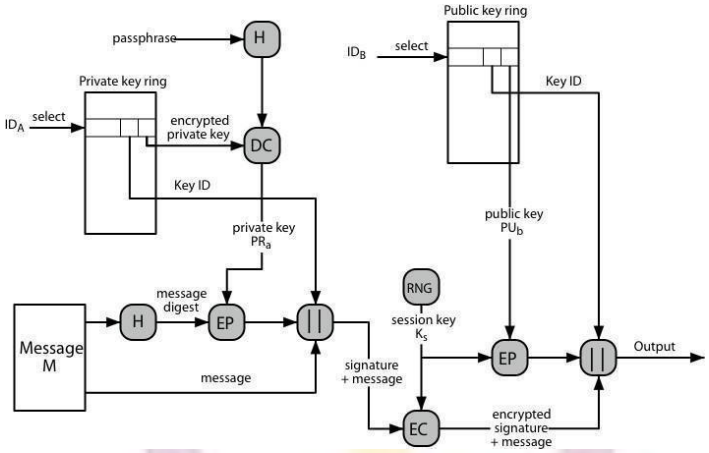
The session key component includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key. The entire block is usually encoded with radix-64 encoding.

Notation:
$E(PU_b, \bullet)$ = encryption with user b's public key
$E(PR_a, \bullet)$ = encryption with user a's private key
$E(K_s, \bullet)$ = encryption with session key
ZIP = Zip compression function
R64 = Radix-64 conversion function

PGP Message Transmission and Reception

Message transmission

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



The sending PGP entity performs the following steps:

Signing the message

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

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

c. The signature component of the message is constructed

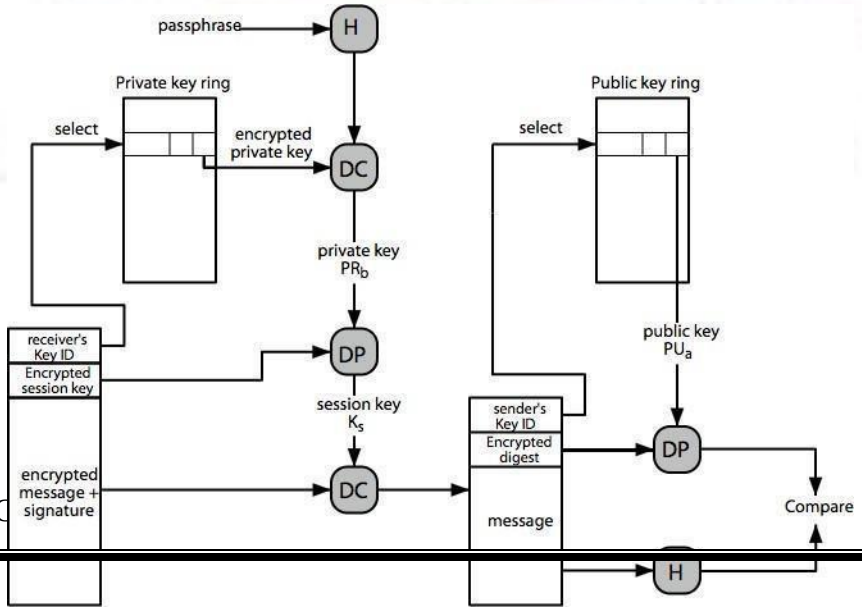
Encrypting the message

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

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

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

Message Reception



The receiving PGP entity performs the following steps:

Decrypting the message

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

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

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

Authenticating the message

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

b. PGP recovers the transmitted message digest.

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

2). S/MIME

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

RFC 822

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

Multipurpose Internet Mail Extensions

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

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

Text data that includes special characters (e.g., Hungarian text) cannotbe transmitted as SMTP is limited to 7-bit ASCII

Some servers reject mail messages over a certain size

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

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

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

The MIME specification includes the following elements:

1.Five new message header fields are defined, which provide information about the body of the message.

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

3.Transfer encodings are defined that protect the content from alteration by the mailsystem.

MIME - New header fields The five header fields defined in MIME are as follows:

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

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

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

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

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

MIME Content Types The bulk of the MIME specification is concerned with the definition of a variety of content types. There are seven different major types of content and a total of 15 subtypes. In general, a content type declares the general type of data, and

the subtype specifies a particular format for that type of data. For the text type of body, the primary subtype is plain text, which is simply a string of ASCII characters or ISO 8859 characters. The enriched subtype allows greater formatting flexibility. The multipart type indicates that the body contains multiple, independent parts. The Content-Type header field includes a parameter called boundary that defines the delimiter between body parts. This boundary should not appear in any parts of the message. Each boundary starts on a new line and consists of two hyphens followed by the boundary value. The final boundary, which indicates the end of the last part, also has a suffix of two hyphens. Within each part, there may be an optional ordinary MIME header. There are four subtypes of the multipart type, all of which have the same overall syntax.

Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript.
	octet-stream	General binary data consisting of 8-bit bytes.

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.

The message type provides a number of important capabilities in MIME. The message/rfc822 subtype indicates that the body is an entire message, including header and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 822 message, but also any MIME message. The message/partial subtype enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this subtype, three parameters are specified in the Content-Type: Message/Partial field: an id common to all fragments of the same message, a sequence number unique to each fragment,

and the total number of fragments. The message/external-body subtype indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. The application type refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-basedapplication.

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

MIME Transfer Encodings

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

The MIME standard defines two methods of encoding data. The Content-Transfer- Encoding field can actually take on six values. Three of these values (7bit, 8bit, and binary) indicate that no encoding has been done but provide some information about the nature of the data. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used, for which a name is to be supplied. The two actual encoding schemes defined are quoted- printable and base64. Two schemes are defined to provide a choice between a transfer technique that is essentially human readable and one that is safe for all types of data in a way that is reasonably compact. The quoted-printable transfer encoding is useful when the data consists largely of octets that correspond to printable ASCII characters. In essence, it represents nonsafe characters by the hexadecimal representation of their code and introduces reversible (soft) line breaks to limit message lines to 76 characters. The base64 transfer encoding, also known as radix-64 encoding, is a common one for encoding arbitrary binary data in such a way as to be invulnerable to the processing by mail transport programs.

Canonical Form

An important concept in MIME and S/MIME is that of canonical form. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in

contrast to native form, which is a format that may be peculiar to a particular system.

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

S/MIME Functionality

S/MIME has a very similar functionality to PGP. Both offer the ability to sign and/or encrypt messages.

Functions

S/MIME provides the following functions:

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

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

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

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

3). IP SECURITY OVERVIEW

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

Need for IPSec

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

Applications of IPSec

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

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

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

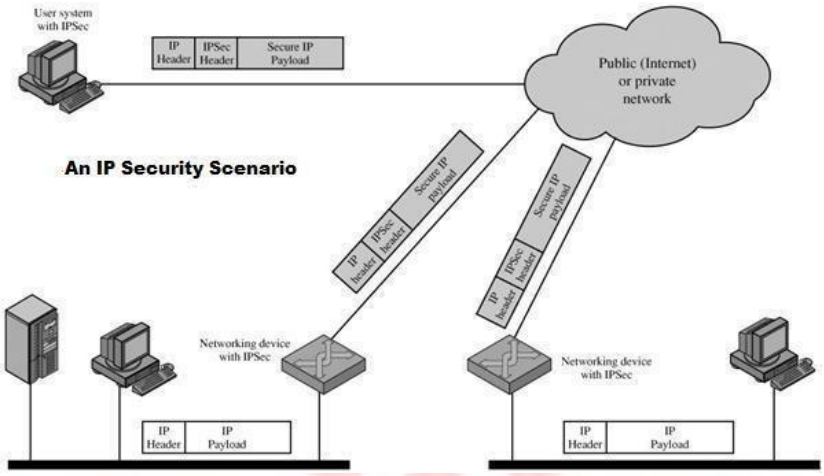
Establishing extranet and intranet connectivity with partners: IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

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

The principal feature of IPSec enabling it to support varied applications is that it can encrypt and/or authenticate all traffic at IP level. Thus, all distributed applications,

including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.

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



Benefits of IPSec

The benefits of IPSec are listed below:

- IPSec in a firewall/router provides strong security to all traffic crossing the perimeter
- IPSec in a firewall is resistant to bypass
- IPSec is below transport layer(TCP,UDP), hence transparent to applications
- IPSec can be transparent to end users
- IPSec can provide security for individual users if needed (useful for offsite workers and setting up a secure virtual subnetwork forsensitive applications)

Routing Applications

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

- router advertisements come from authorized routers
- neighbor advertisements come from authorized routers
- redirect messages come from the router to which initial packet was sent
- A routing update is not forged

4). IP SECURITY ARCHITECTURE

To understand IP Security architecture, we examine IPSec documents first and

then move on to IPSec services and Security Associations.

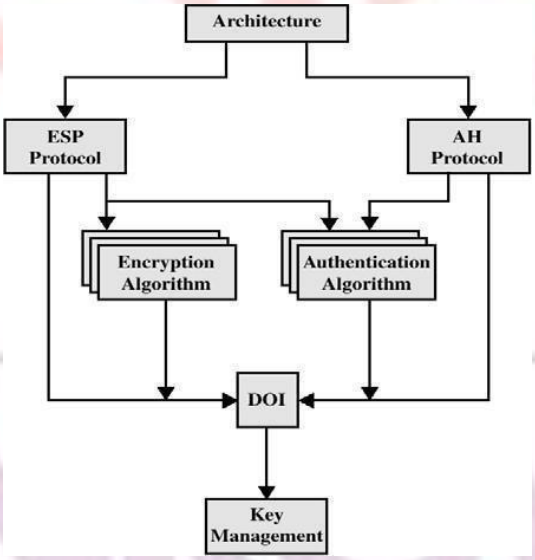
IPSec Documents

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

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

Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header.

The extension header for authentication is known as the Authentication header; that for encryption is known as the Encapsulating Security Payload (ESP) header. In addition to these four RFCs, a number of additional drafts have been published by the IP Security Protocol Working Group set up by the IETF. The documents are divided into seven groups, as depicted in following figure



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

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

IPSec Services

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

Connectionless Integrity:- Data integrity service is provided by IPSec via AH which prevents the data from being altered during transmission.

Data Origin Authentication:- This IPSec service prevents the occurrence of replay attacks, address spoofing etc., which can be fatal.

Access Control:- The cryptographic keys are distributed and the traffic flow is controlled in both AH and ESP protocols, which is done to accomplish access control over the data transmission.

Confidentiality:- Confidentiality on the data packet is obtained by using an encryption technique in which all the data packets are transformed into ciphertext packets which are unreadable and difficult to understand.

Limited Traffic Flow Confidentiality:- This facility or service provided by IPSec ensures that the confidentiality is maintained on the number of packets transferred or received. This can be done using padding in ESP.

Replay packets Rejection:- The duplicate or replay packets are identified and discarded using the sequence number field in both AH and ESP.

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

5). AUTHENTICATION HEADER

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

IPSec Authentication Header

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

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

Reserved (16 bits): For future use.

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

Sequence Number (32 bits): A monotonically increasing counter value, discussed later.

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

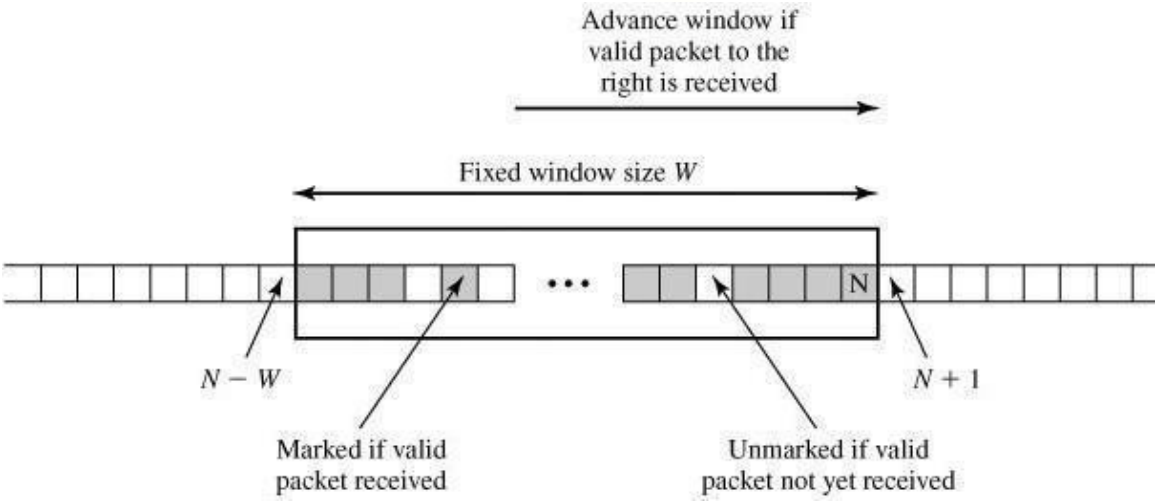
Anti-Replay Service

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

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

The IPSec authentication document dictates that the receiver should implement a

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



1.If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.

2.If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.

3.If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

Integrity Check Value

ICV is the value present in the authenticated data field of ESP/AH, which is used to determine any undesired modifications made to the data during its transit. ICV can also be referred as MAC or part of MAC algorithm. MD5 hash code and SHA-1 hash code are implemented along with HMAC algorithms i.e.,

- HMAC-MD5-96
- HMAC-SHA-1-96

In both cases, the full HMAC value is calculated but then truncated by using the first 96 bits, which is the default length for the Authentication Data field. The MAC is calculated over

IP header fields that either do not change in transit (immutable) or that are predictable in value upon arrival at the endpoint for the AH SA. Fields that may change in transit and

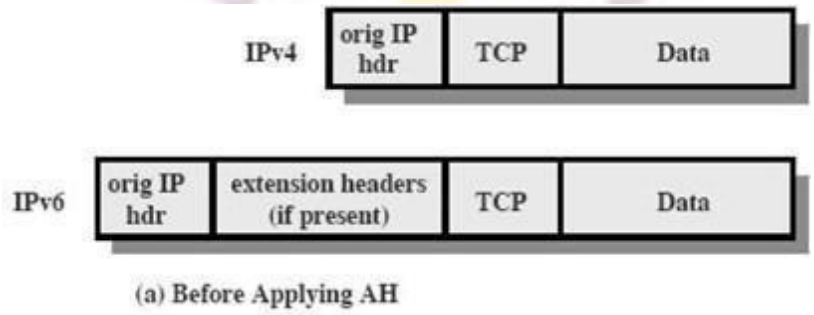
whose value on arrival is unpredictable are set to zero for purposes of calculation at both source and destination.

The AH header other than the Authentication Data field. The Authentication Data field is set to zero for purposes of calculation at both source anddestination.

The entire upper-level protocol data, which is assumed to be immutable in transit (e.g., a TCP segment or an inner IP packet in tunnel mode).

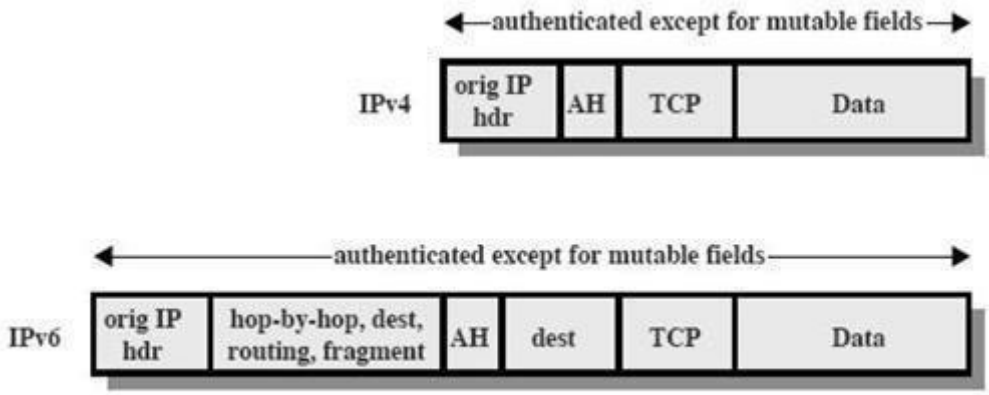
Transport and Tunnel Modes

The following figure shows typical IPv4 and IPv6 packets. In this case, the IP payload is a TCP segment; it could also be a data unit for any other protocol that uses IP, such as UDP or ICMP.

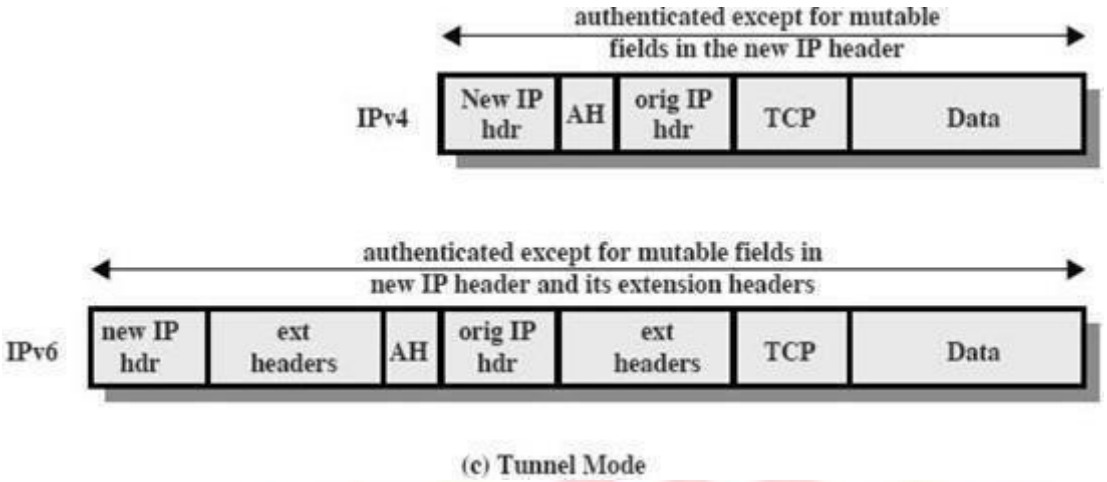


(a) Before Applying AH

For transport mode AH using IPv4, the AH is inserted after the original IP header and before the IP payload (e.g., a TCP segment) shown below. Authentication covers the entire packet, excluding mutable fields in the IPv4 header that are set to zero for MAC calculation. In the context of IPv6, AH is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the AH appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the AH header, depending on the semantics desired. Again, authentication covers the entire packet, excluding mutable fields that are set to zero for MAC calculation.



(b) Transport Mode



(c) Tunnel Mode

For tunnel mode AH, the entire original IP packet is authenticated, and the AH is inserted between the original IP header and a new outer IP header. The inner IP header carries the ultimate source and destination addresses, while an outer IP header may contain different IP addresses (e.g., addresses of firewalls or other security gateways). With tunnel mode, the entire inner IP packet, including the entire inner IP header is protected by AH. The outer IP header (and in the case of IPv6, the outer IP extension headers) is protected except for mutable and unpredictable fields.

6). ENCAPSULATING SECURITY PAYLOAD

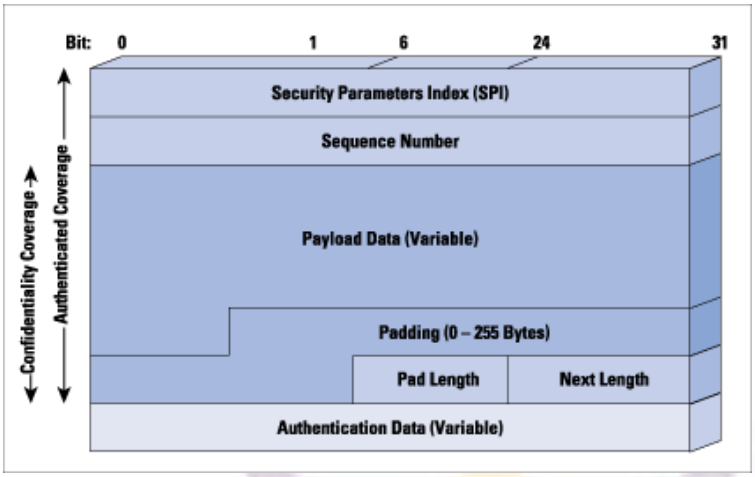
The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format

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

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

Sequence Number (32 bits): A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.



Payload Data (variable): This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.

Padding (0-255 bytes): This field is used to make the length of the plaintext to be a multiple of some desired number of bytes. It is also added toprovideconfidentiality.

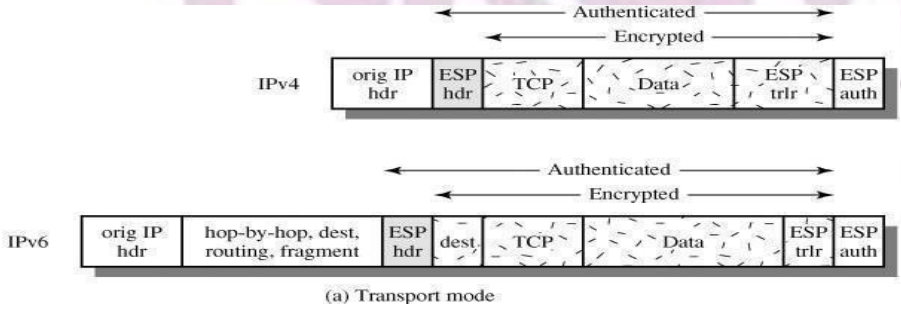
Pad Length (8 bits): Indicates the number of pad bytes immediately preceding thisfield.

Next Header (8 bits): Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).

Authentication Data (variable): A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

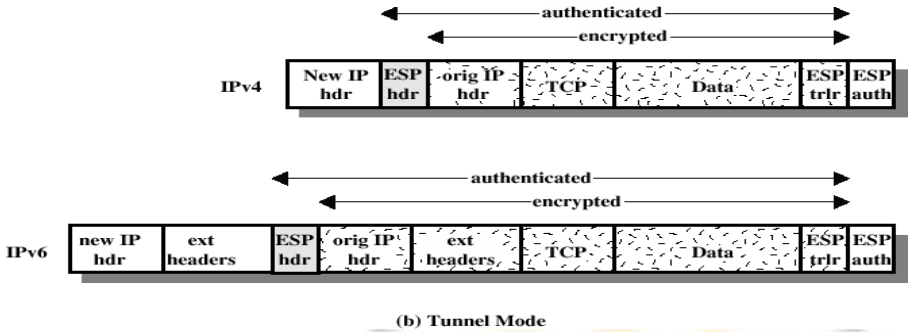
Adding encryption makes ESP a bit more complicated because the encapsulation *surrounds* the payload rather than *precedes* it as with AH: ESP includes header and trailer

Transport Mode ESP



(a) Transport mode

Tunnel Mode ESP

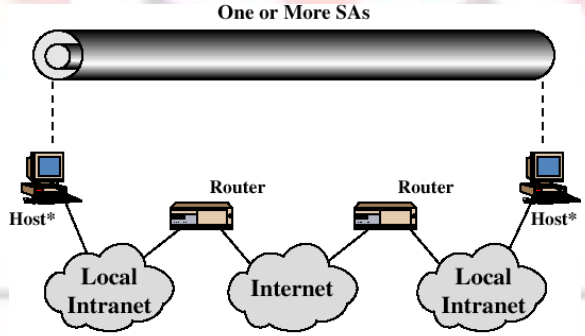


(b) Tunnel Mode

Basic Combinations of Security Associations

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

Case:-1



(a) Case 1

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

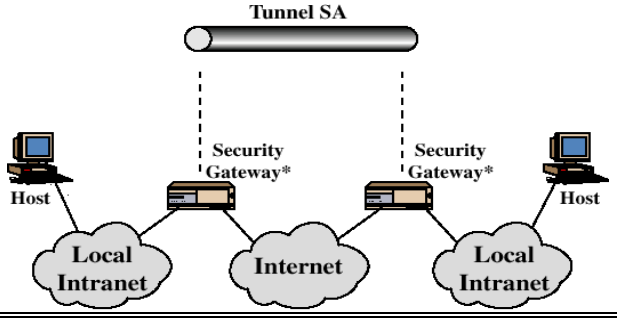
a)AH in transport mode

b)ESP in transport mode

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

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

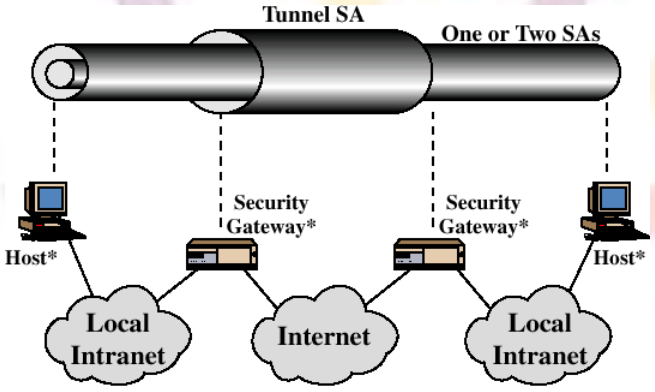
Case:-2



(b) Case 2

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

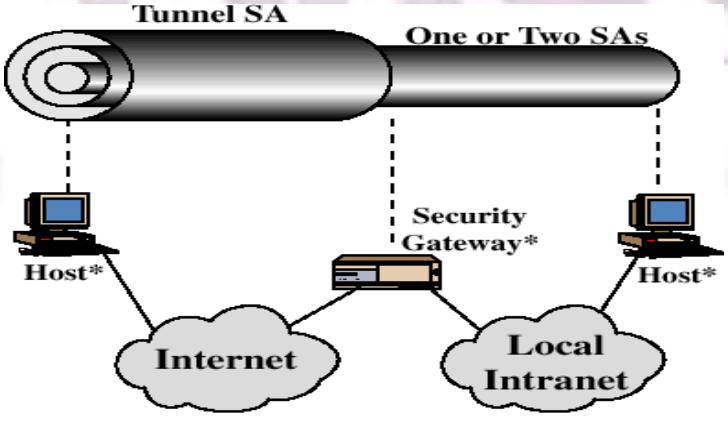
Case-3:-



(c) Case 3

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

Case:-4



(d) Case 4

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

7). COMBINING SECURITY ASSOCIATIONS

Since IPSEC is designed to be able to use various security protocols, it uses Security Associations (SA) to specify the protocols to be used. SA is a database record which specifies security parameters controlling security operations. They are referenced by the sending host and established by the receiving host. An index parameter called the Security Parameters Index (SPI) is used. SAs are in one direction only and a second SA must be established for the transmission to be bi-directional. A security association is uniquely identified by three parameters:

Security Parameters Index (SPI): A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

IP Destination Address: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.

Security Protocol Identifier: This indicates whether the association is an AH or ESP security association.

SA Parameters

In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:

Sequence Number Counter: A 32-bit value used to generate the Sequence Number field in AH or ESP headers

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

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

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

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

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

IPSec Protocol Mode: Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section.

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

Transport and Tunnel Modes

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

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

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

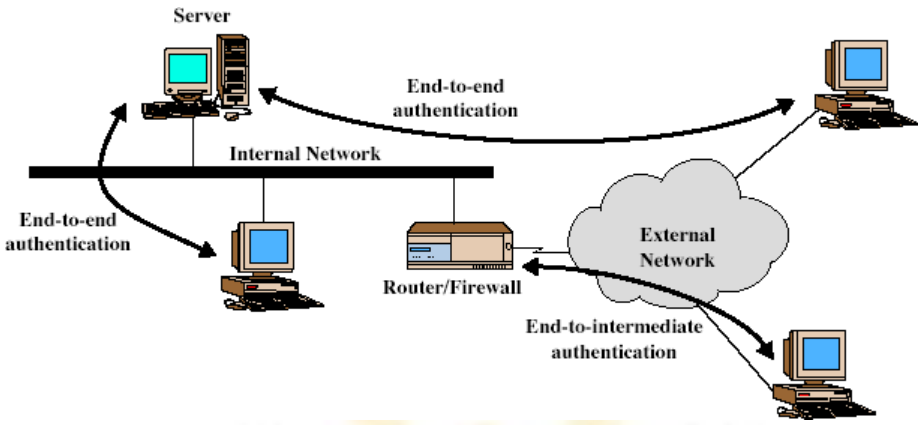
a.Transport mode: the IP sec header is inserted just after the IP header –this contains the security information, such as SA identifier, encryption, authentication

Typically used in end-to-end communication IP header not protected

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

Typically used in firewall-to-firewall communication Provides protection for thewhole IP packet

No routers along the way will be able (and will not need) to checkthe content of the packets.



8). INTERNET KEY EXCHANGE

What Does Internet Key Exchange (IKE) Mean?

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

IKE is a hybrid protocol based on:

A.ISAKMP (RFC2408): Internet Security Association and Key Management Protocols are used for negotiation and establishment of security associations. This protocol establishes a secure connection between two IPSec peers.

B.Oakley (RFC2412): This protocol is used for key agreement or key exchange. Oakley defines the mechanism that is used for key exchange over an IKE session. The default algorithm for key exchange used by this protocol is the Diffie-Hellman algorithm.

C.SKEME: This protocol is another version for key exchange.

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

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

Techopedia Explains Internet Key Exchange (IKE)

The IKE works in two steps. The first step establishes an authenticated communication channel between the peers, by using algorithms like the Diffie-Hellman key exchange, which generates a shared key to further encrypt IKE communications. The communication channel formed as a result of the algorithm is a bi-directional channel. The authentication of the channel is achieved by using a

shared key, signatures, or public key encryption.

There are two modes of operation for the first step: main mode, which is utilized to protect the identity of the peers, and aggressive mode, which is used when the security of the identity of the peers is not an important issue. During the second step, the peers use the secure communication channel to set up security negotiations on behalf of other services like IPSec. These negotiation procedures give rise to two unidirectional channels of which one is inbound and the other outbound. The mode of operation for the second step is the Quick mode.

IKE provides three different methods for peer authentication: authentication using a pre-shared secret, authentication using RSA encrypted nonces, and authentication using RSA signatures. IKE uses the HMAC functions to guarantee the integrity of an IKE session. When an IKE session lifetime expires, a new Diffie-Hellman exchange is performed and the IKE SA is re-established.

Case Studies on cryptography and Security

9). SECURE MULTIPARTY CALCULATION

In the last few decades, data privacy and security has become the primary concern to everyone. Due to the rise in technological advancements and the internet, it has been a challenging task to provide data security and data privacy of the data, when data is distributed over large distributed networks. As everyone is now concerned with their data, a lot of research is going on how to provide data security and privacy to the participants in the network. One of the techniques which provide the solution to the problems of data security and data privacy is **Secure Multiparty Computation.** The secure multiparty computation may be defined as the problem of 'n' players to compute jointly on an agreed function securely on the inputs without revealing them.

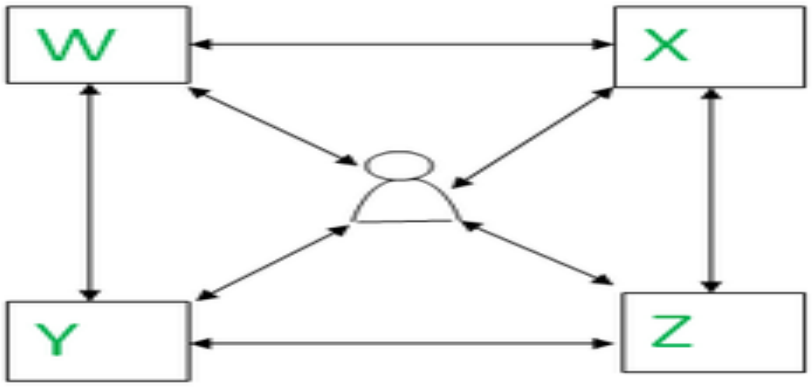
History

Secure multiparty computation started early in the 1970s. It was known as multiparty computation at that time. It does not gain popularity at that time as it was not implemented practically. In the 1982's it was introduced as secure two-party multiparty computation. It is used to solve a lot of problems of computation without revealing the inputs to other parties. Finally, it came with a name as secure multiparty computation in which the functions of different types are computed, that is the reason it is sometimes called **SFE- Secure Function Evaluation**.

1.The secure multiparty computation is used for the utilization of data without compromising privacy.
2.It is the cryptographic subfield that helps in preserving the privacy of the data.
3.Emerging technologies like blockchain, mobile computing, IoT, cloud computing has resulted in the rebirth of secure multiparty computation.
4.Secure multiparty computation has become the hot area of research in the last decade due to the rise of blockchain technology.
5.The researchers are now more interested to implement secure multiparty computation in distributed systems.
6.Unlike in centralized systems, secure multiparty computation may have better performance in distributed systems.

Architecture

The secure multiparty computation provides a protocol where no individual can see the other parties data while distributing the data across multi parties. It enables the data scientists and analysts to compute privately on the distributed data without exposing it.



Multiparty sharing data among each other with any third party using a specified protocol.

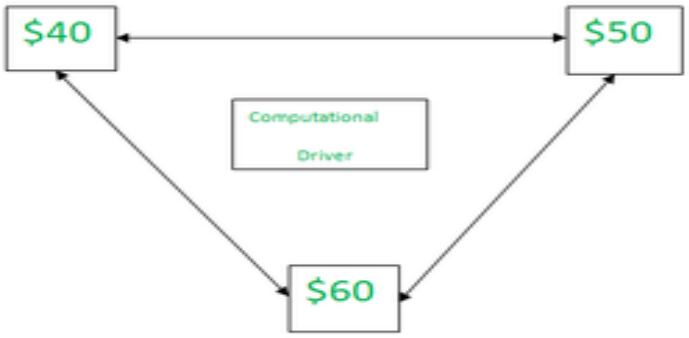
The co-workers want to compute the maximum salary without revealing their individual salary to others. To perform such a computation, secure multiparty computation is implemented to calculate the maximum salary. The parties in a distributed manner jointly perform a function to calculate it without revealing the salary. Data in use is kept in encrypted form, broken up, and distributed across parties, there are no chances of quantum attacks. It is impossible to have a trusted party in the real world, as all parties communicate with each other in one or the other way In such a scenario, the parties may get corrupted. The corrupted parties have behavior like semi-honest and malicious.

1.A semi-honest opponent is one who follows the specified protocol but makes the parties corrupted. The protocol is run honestly, but they try to extract information from the messages exchanged between parties.

2.A malicious adversary makes an attempt to breach security and does not follow the specified protocol. The adversary can make the changes during the execution process of the protocol. While using multiparty computation, we assume the party is honest which follows all the protocols.

Example

Suppose we want to compute the average salary among three employees without revealing the actual salary, for such problems one can use secure multiparty computation. Let's take an example-



Example of computing average salary of multiparty using additive sharing.

Mathematical representation of the problem can be given as:

F(A, B, C) = Average (A, B, C)

Sam, Bob, and Cassy want to calculate their average salary.

1. Say Sam's salary is $40k. Using additive sharing, $40k is split into randomly generated three pieces $44k, $-11k, and $7k.
2. Sam keeps one of these secret pieces with herself and distributes the other two to each.
3. The same procedure is followed by all three.
4. Secret sharing keeps the data in encrypted form when in use. The procedure is given below-

Sam	Bob	Cassy	
44	-11	7	**$40**
-6	32	24	**$50**
20	0	40	**$60**
$58	**$21**	**$71**	

Total salary = $150

Average Salary = 150/3

= $50

From the above data shared there is no clue about the actual salary, but the average salary is being calculated.

Techniques

There are a number of techniques developed for secure multiparty computation protocol construction having different features. Some techniques used in secure Multiparty computation are listed below:

1. **Shamir Secret Sharing:** Secret sharing is utilized as the basic tool when there is an honest majority in secure multiparty computation. A secret sharing scheme is that a secret s is shared among n parties, such that t+1 or more parties come together to reconstruct the secret. The parties lesser than t cannot get any information or reconstruct the secret. The scheme which fulfills the requirements of t+1 out of n is called the threshold secret sharing scheme.
2. **Honest Majority MPC:** The function can either be represented by Boolean or arithmetic circuit in an honest majority. For MPC-based secret sharing having the honest majority, there is finite field Zp with p>n for arithmetic circuit and the circuit is Turing complete.
3. **Input sharing:** Every party shares the input using the Shamir secret sharing. The circuit is being provided the input for computation. Every party keeps his input private by adding some random number to the input and finally, after getting the output the random number is known to the party is removed, and we get the output.
4. **Circuit evaluation:** The circuit is evaluated by parties one gate at a time. The gates are evaluated serially from input to output. The evaluation consists of the computation of addition and multiplication gates. For inputs a(x) and b(x), the output of addition for the ith party is calculated as c(i) = a(i) + b(i). Similarly, the output of multiplication for the ith party is calculated as c(i) = a(i) . b(i).
5. **Private set intersection:** The private set intersection protocol is very efficient for the two parties' problems. Two parties who wish to find the elements of intersection with private set of inputs without revealing the input, the private set intersection is better approach for both honest and dishonest adversaries.
6. **Threshold cryptography:** Threshold cryptography aims to carry out the cryptographic operations for a set of parties without holding the secret by any of the single party. RSA algorithm is used for the scheme where the basic function is y=xe mod n. RSA is used for encrypting secrets or messages.
7. **Dishonest majority MPC:** In the secure multiparty computation, there can be both honest and dishonest parties. The secure Multiparty computation is secure as long as there is an honest majority. If the adversaries are corrupt more than the majority, new approaches are required for security. For the dishonest majority, there are protocols like GMW oblivious transfer, garbled circuit, Tiny oz and many more protocols.

Benefits Of Secure Multiparty Computation

Let's discuss some benefits of secure multiparty computation:

1. **Trusted third party:** In Secure Multiparty Computation, we can share data in a distributed manner with different organizations without any third party and even the privacy of data will be preserved while sharing data.
2. **Data Privacy:** The private data of organizations can be shared for computation purposes. The concern of data privacy is provided by using secure multiparty computation, which keeps the data in use in encrypted form. Thus, the data is not revealed or compromised.
3. **High accuracy:** Secure Multiparty Computation provides highly accurate results for different computations using cryptography.
4. **Quantum safe:** The data shared between parties is safe against quantum attacks, as the data is broken up and encrypted when distributed among parties for computation.

Limitations Of Secure Multiparty Computation

Secure multiparty computation being used for solving different problems, but there are few limitations. The main limitations are the computational overhead and high communication costs.

1. **Computational overhead:** To provide the security we need to generate the random numbers, the random number generation requires more computation overhead which slows down runtime.
2. **High communication costs:** Distributing the data to multiple parties for computation over the networks leads to higher costs of communication.

10). VIRTUAL ELECTIONS

INTRODUCTION

Elections are held everywhere. However, voters have to go to the polling station to cast their vote. The political membership process is exceptionally complex and many things are needed to advance voting. Large arrangements have been made to finish. It involves manual work. Government elections are held by area. To vote, the voter must be available to vote at the polling place. This may reduce voter support; Web-based voting simplifies this undertaking. Voting in Cryptography involves security and a secure system. It is important to implement such a system. This will reduce labor, make ballet easier to use and more productive. Individuals must be available at the location for selection. Cryptography is a system of encoding voter details. In this system, the client will be contacted to upload a security message and voter details during registration. The customer will receive the security part of the security picture via email. This share will be in an encrypted format. The customer can log in to the system to change the details at any time. Only when voting, the customer must upload a security share. If the share is incorrect, the poll cannot be voted on, because the security share is generated using random pixels, so the real picture cannot be predicted. Additionally, the share cannot be retrieved by some other client or disapproved person as it will be securely sent via email. Ballet casting will only be fruitful if the correct share relating to that client is uploaded.

Fraud sends fake messages or sets up fake sites that copy. Phishing is a form of identity online identity theft in which fraudsters manipulate Internet users to submit personal information to illegal websites. Phishing tricks are usually displayed as spam or pop-up and are always difficult to identify. When fraudsters obtain your data, they can use it for all kinds of identity fraud, risking your great reputation and a great name. Fishers will become more sophisticated in the design of their fake sites. Phishing is the data of the types of fraud, so be comfortable with a variety of phishing tricks for you as well as figure out how to prepare for it. The most valid and direct way to secure a system asset is to assign it a unique name and a corresponding password.

Cryptography is the study of protecting data. It has been used as a means of safe communication between people and governmental organizations. Today, cryptography is the foundation of advanced security technologies used to secure data and assets on both open and closed networks. Belief is the process of examining the personality of a person or thing. When you confirm something, the purpose is to check that you have a real deal. It is necessary to implement their methods to determine the level of authorization of the user of the application. Applications often do this by keeping private records that include the names of customers to whom who has access. Databases applications, for example, regularly maintain private approval tables to control the fields in records that a particular client can view or modify. Few people advocate the benefits it brings, for example, mobility, openness, improved speed and accuracy in the delivery of ballots from home and the same number that it represents are concerned about the crisis, for example, inconsistent entry, breach of mystery, and ambiguity. And a change in the effect of a political race. The project focuses on the prevention of phishing attacks and secure authentication of Internet voting systems using cryptography. Cryptography is an encryption strategy to hide data so that it can be decrypted by human vision if the right key picture is used. Cryptography is the study of protecting data. It has been used as a means of safe communication between people and governmental organizations. Today, cryptography is the foundation of advanced security technologies used to secure data and assets on both open and closed networks. Belief is the process of examining the personality of a person or thing. When you confirm something, the purpose is to check that you have a real deal. It is necessary to implement their methods to determine the level of authorization of the user of the application. Applications often do this by keeping private records that include the names of customers to whom who has access. Databases applications, for example, regularly maintain private approval tables to control the fields in records that a particular client can view or modify. Few people advocate the benefits it brings, for example, mobility, openness, improved speed and accuracy in the delivery of ballots from home and the same number that it represents are concerned about the crisis, for example, inconsistent entry, breach of mystery, and ambiguity. And a change in the effect of the political race. The project focuses on the prevention of phishing attacks and secure authentication of Internet voting systems using cryptography. Cryptography is an encryption strategy to hide data so that it can be decrypted by human vision if the right key picture is used.

2.LITERATURE REVIEW

From the time it takes to the current technological development, there are online voting systems. That was clarified in this document. Develop voting plans to make more efficient voting services avalable with ICT resources than traditional paper-based voting methods. Voters regard themselves as consumers and it is expected that the government will make the voting business more convenient. In the past decade, various forms of electronic voting, especially as additional methods of voting for remote voting, political parties, candidates, the electoral administration, and most importantly to

improve the efficiency and promise of the democratic process to the electorate have attracted considerable attention.

It allows voters to access the public algorithm and parameters to confirm their turnout.

Three types of voting systems exist:

1.System of paper voting

The paper voting system is the most common system for voting. Before the electronic voting system is implemented, it will be used. The system of paper ballet includes paper and sealed ballet. Each voter uses and does not share one ballot. This system's disadvantages are

i) the time it takes;

ii) the speed is low.[16]

2.Electronic voting system

Electronic voting systems are electronic voting devices. A voting machine that uses an electronic voting machine to allow voters to pass on their secret ballots. The inconvenience is I poor computer science individuals cannot vote correctly, (ii) safety threats sensitive, (iii) electricity consumption at polling stations; and (iv) costs.

3.Online voting system

A new platform for secure votes and voting is the online voting system. Online voting systems are a web-based voting system, which transmits votes via a web browser over the internet. Voters from all over the world are eligible to vote online.

Security issues arising from online voting are as follows: In general applications, password protection is high and phishing attacks are not the focus of the application. Website users are not protected efficiently from phishing.

The key proposal for ensuring a secure online polling protocol to meet privacy, anonymity, eligibility, equity, verification, and unique online voting safety requirements

To achieve reliability, eligibility, transparency, accuracy, and uniqueness of the e-vote system, two milliardaires couples have created secure online voting for identities based on cryptographic algorithms.

A secure, end-to-end verifiable, Identity-based blind signature Internet voting system: IEEE, newspapers, 2020; This document has been amended Early vote, elliptical curve cryptography, verifiable end-to-end digital signature, Internet vote system. Batch venerability. Functional digital signature used by the BLS short signature system to protect voting against any changes anonymously

to issue a blank ballot to voters. Future of voting: Specifications and feasibility study of verifiable Internet vote from end to end.

Phish-haven-An Efficiency Real-Time AI Phishing URLs Detection System: IEEE, newspapers, 2020; This article changed phishing URLs generated by AI, machine learning, phishing URLs created by people, lexical features, multi- threads, HTML URL encoding. Extracts web pagecontent which is therefore ineffective in computation. Non- proactive method Needs source codes or the website's entire website content. The use of multiple threading technologies on an input unit and output unit may be further enhanced by the incorporation of unattended learning.

SeVEP: Electronic polling system secure and verifiable: 2019 IEEE, journals, Authentication modified, efficiency, electronic polling, malware, security, compliance. Authentication, electronic polling process has resource allocation polling system. Developing a working SeVEP prototype and assessing its scalability and usability for real-world use.[13]

Towards Developing a Secure and Robust Solution for E-Voting using Block-chain: 2019 IEEE, Spring, This paper modified coercion resistance problem, Blockchain, Online Voting process, Developing a Secure Solution for online Election process information and To solve coercion resistance problem to solve using cryptographic algorithms.[18]

End to End Verifiable Electronic Voting System for Shareholders: IEEE 2019, newspaper, this article amended Electronic vote, voting by shareholders, verification end-to- end, zero evidence of knowledge, Decision Diffuse the assumption by Hellman, safety evidence and verifiable electoral process. More generally, voters can leave and leave dynamically within calculation periods if using a smartphone.

Secure Online Voting System Using VC: 2018 IEEE, Spring, this paper modified and using Visual cryptography, security share, voting system. Secure a voting process for using Cryptography task scenario and Improvement in an existing algorithm.

A Scheme for Three-Way Secure and Verifiable E-Voting: 2019 IEEE, journal, This paper modified and using Electronic Voting, Anonymity, Verifiability, and Paillier Cryptosystem, Homomorphic Encryption process on the distributed implementation of Three way Secure and Verifiable Election process.[21]

The Security Issues of The Online Voting System: While inheritance of such items in the source code is not acceptable, the root of the security problems which have occurred have not only been attributed to outsiders (for example voters and attackers) but also to insiders (for example program developers and administrators). These mistakes caused a vote system crash.

The solutions suggested for stopping these attacks have therefore been outlined. To prevent hackers from getting into the voting system over a network we can, for example, develop our system to transmit data without a network. Another example is to limit voting to unique input data to prevent command injection.

4.METHODOLOGY

The rapid development of technologies and Internet popularity lead to the digitization of diverse types of technology, such as electronic commerce, e-democracy, e- government, etc. To minimize costs and red tape in public departments, the contemporary states are seeking to provide people who can participate and benefit from online services by increasing the number of activities associated with this new medium. Electronic voting is one of the most important Internet-related activities. The modern recently We consider the same methodology as the one we discussed for estimating the operating machine cycles (for example, private and public operations based on Salsa20 algorithm, operations on elliptic curve and pairing).

For example, (1) use of electronic voting can reduce or eliminate undesirable human errors, (2) in addition to its reliability, the online voting system does not need geographical proximity of voters which increase the number of participating voters, (3) e-voting saves a lot of time for voters and reduce a cost when counting the voted ballots.

a.What approach is taken by the author

Once all the nodes of the network are running, a new user can connect to the server. The user registers a non-anonymous user (using Adhar Card, phone, password, etc), and performs the login. The user produces an RSA key pair locally (private key & public key). With the Public-Key server, the user blinds his public key. The public key of the user is blinded and forwarded to the server.[16]

The server Blind Signs the Public-Key blinded from the user and returns it to the user. The user unbinds the Public-Key signed by the server, and now has the Public-Key Blind Signed by the server. The user sends the Public-Key blind signed to the p2p network. The peers verify that the Public- Key Blind Signed is correctly signed by the server, if it is, they add the Public-Key to the Ethereum Blockchain, inside a new block.

b.Our approach

As per recent research RSA method to secure data with blind signature has some flaws and can be cracked using high-end computational devices. So we will be using a more secure Salsa20 security algorithm which is found more to be more secure than an existing algorithm like RSA and AES. Also, Salsa20 is more FAST and lightweight than RSA and AES. Salsa20 is FAST in terms of encrypting and decrypting. This means it can encrypt more messages per cycle compare to RSA and AES. Also, it is lightweight means it requires less computational resources compared to others. Despite such benefits, Salsa20 provides better security.

5.REQUIREMENT ANALYSIS

Before designing a voting system, a complete and detailed set of requirements must be developed. The design requirements for the online voting system are divided into 2 groups during this work: the general one and the system one. The general requirements of any voting system are complied with. The requirements of a system are, on the other hand, essential for the development of a developed

system. System-specific requirements, on the other hand, are system-specific demands. Allow system requirements specific to the system:

i.Multi-user: Many voters can vote simultaneously;

ii.Accessibility: System access can be accessed by voters in any location using secure internet and/or mobile devices.

iii.Design of the system framework:

iv.The framework was designed to define the frameworks for the application. The structure for the defined objective is the emerging framework of this design process. The infrastructural model architecture in which models are developed is an integral component of the model design.

v.Based on the earlier (the study was not published), in comparison to cryptography, it can be seen that the cryptographic algorithms of voice data packets using serpent damage or loss of some packages during shipping. No Voice Data Packet Loss occurs when you push to talk to the algorithm salsa20.

vi.And the salsa20 algorithm in another previous study Implementing the security and SMS is found to be relatively short in Salsa20 encryption and decryption.

vii.This experimental test enables analysis of salsa20 Stream cipher algorithm as a cryptographic sound data packet algorithm. From Table 1. we can see that the first packet encryption process is Salsa 20 Faster than the decryption process.

viii.When the Salsa20 algorithms are being implemented to speak, there is a delay of 1.9 seconds, but the push to talk application doesn't change performance.

ix.Encryption & decryption of voice data Packs is successful because the encrypted voice data packets on the Android Smartphone can be heard using Salsa20 algorithms.

x.The bits Modified from the bits of the normal audio data packet with bits of the encrypted audio data packets can be seen from an avalanche effect test. We know that salsa20 has good performance to secure voice data packets based on the Avalanche effect's value.[19]

6.SOFTWARE REQUIREMENT

To test this framework, the software has been developed and deployed. The program is based on Java, Spring Tool, XAMPP server, HTTP SMS gateway. Windows XP, Windows10, and others.

Evaluation and performance checks:

User understanding of the system is developed following experimental use to determine if the core values required in the voting system have been developed in accordance with the online voting system.

The following research questions arose in connection with guided questions whether the developed online voting system meets the desired general safety requirements of voting systems:

a.Can a vote be unreserved? "Integrity" requirement,

b.Is it possible to verify who electors claim to be? "Authenticity" requirement.

c.Is it possible to vote only once by eligible voters through the developedonline voting system? "Democracy Requirement."

d.Can no polling be ensured by the developed online voting system.

liked to the electorate or any other voter? Requirement for Privacy.

7.SYSTEM DESIGN

The system has three modules such as:

1.admin module,

2.client module and

3.server module.

The application requires the user to register and then register with the same username and password. The user must then choose the candidate to vote for. After the user clicks on the

'Vote' buttons, his e-mail id will be transmitted to him, whereas share2 will be downloaded automatically on the server.

Certified users will be shown a captcha that users must use properly. Properly entered into the voter details, the users vote will be successfully registered.

To approximate the computer cycles consumed by operations using a highly verifiable safe online voting system in which each elector is authenticated using a unique identifier provided by the relevant authority and his biometric details (for example, private and public operations based on RSA, operations on elliptic curve and pairing). The appropriate cryptographic operations' notations and the number of computer cycles they absorb. Our system's success in comparison to other systems. The suggested method and system are based on ECDL and GDH problems and use elliptic curve cryptography. The security of the systems is focused on the discrete logarithm problem (DLP) and integer factorization problems, and they are based on the standard RSA public-key cryptosystem (IFP). The ECC-based operations (scalar multiplication and addition) are more efficient than the RSA-based operations.

4.Requirements for an election system

Researchers also defined a series of specifications for a reliable electronic voting protocol in this framework.

5.Security Requirements

Since the internet seems to be an unstable place, protection plays an essential role in every voting method, particularly e- voting. For the electronic voting framework to function without bugs, it must be applied according to safe design. Despite the system's difficulty of architecture and execution, it seems that certain principles are universally agreed upon as the minimum security specifications for electronic voting.

6.System-Wide requirements

In this section, the system-wide requirements for implementing voting protocols are discussed-Voter conveniently: Voters should be able to vote without consulting the voting authorities and complete the voting procedures with the bare minimum of skills and equipment- Voter mobility: Voters should be able to vote from either location without restriction. The scheme is successful if the number of electors and the authorities involvement in the protocols is equal to the computing and communications resources.

Salsa20 is Daniel J. Bernstein's software-oriented stream cipher. The algorithm can support 128-bit and 256-bit keys. The updated state is used as a 512-bit keystream output following r iterations of the Salsa20/r round function. Each output block is an independent key, nonce, and counter combination and, since there is no link between blocks, the

Salsa20 operation is similar to a block cipher operation in counter mode.

Analysis: Salsa20 underwent significant cryptographic analysis over the years following its publication. While several attacks on smaller versions of the cipher have been found, there is no better attack than an exhaustive key search on either of the Salsa20.

STREAM CIPHER WITH SYMMETRIC SECRET KEY

Key length = 32 bytes

8.THE SYSTEM ARCHITECTURE

For found detection and prevention, we are proposing a new method for detecting phishing websites. Our method uses cryptography and it is based on an Anti-phishing providing authentication scheme. The proposed system can be divided into two Flowcharts one is Registration Flowchart and the second is Login Flowchart.

1.Registration Flowchart

The text of these images is the password for the user when the flowchart is registered. The image is divided between the user and the server. The image is shared. The user will share the user with the login Flowchart for further verification. The detals of voters are also stored as confidential data for the current website database.

2.Login Flowchart

The first user to request a user name (user id) in the login phase is a user name. Then the users requested to enter their share with him. This share is forwarded to the server where every user stays the share and share stored in the website dataset for the security of each user.

Authentication is the process by which the person claims to be. The user id is sent to the server for this purpose and the appropriate password will be retrieved from the database. Now you compare the password of the user and the password of the database. Thus you can check whether the website is a real/secure website or a phishing website using the username or password generated by stacking two shares and also check whether the user is authenticated or not.

3.Datasets

The online voting system uses an online voting database consisting of two datasets:

Login details The table contains registered user/voter logs and passwords with appropriate user names. It has voters/user contacts, telephone, and e-mail addresses as well.

Voting details – The candidate record and the voters who voted for the candidate shall be included. Its main key is the ID field that is also necessary for counting votes.

8.PERFORMANCE ANALYSIS

This compares our system and its performance with the related online voting systems. Assume that Weil is defined by the PBC library's Type-F (BN curve), 256-bit-group pairing, and 512-bit embedded pairing with RSA-identical security level.

Implementing and comparing the performance of our proposed online voting system to the corresponding machine cycle schemes obtained through simulation. We consider estimation of the operational machine cycles (for example, private and public operations based on curve and pairing, RSA operations on elliptic).

These systems are based on the traditional public-key system Salsa20 based on the discrete logarithm (DLP) system and are safe (IFP). Compared to Salsa20 it can be seen that the ECC (scalar multiplication and supplement) operation is efficient.

9.CONCLUSION AND FUTURE WORK

The cryptographic online voting system overcomes restrictions on the traditional voting system. This system offers more safety and takes a while. There is also no chance of voting fraud. There is a significant reduction in the money spent on security. This method aims primarily to provide full privacy to voters and to ensure that the online voting system is coordinated optimally. The fundamental idea of this system is to use a strong voting authentication security mechanism. Visual encryption encrypts information and can decrypt it without mathematical calculations. People with an internet

connection at home can vote at the polls without any problems. Visual cryptography is used to conduct elections fairly easily and efficiently using these internet-based voting systems since voters can vote from the point of view in which they operate using the online voting system. Various advantages include low costs and increased voting attendance online voting. Online voting offers This online voting system takes careful account of safety and human factors, and in particular, ensures that the electorate has reliable and intuitive indications on the validity of the vote. The system we proposed to provide voters with mutual authentication and choosing with visual encryption.

11). SINGLE SIGN ON

What is single sign-on?

Single sign-on (SSO) is an authentication scheme that enables users to log in to a session once, using a single set of login credentials, and gain secure access to multiple related applications and services during that session without logging in again.

SSO is used commonly to manage authentication in company intranets or extranets, student portals, public cloud services, and other environments where users need to move between multiple applications to get their work done. It's also used increasingly in customer-facing web sites and apps – such as banking and e-commerce sites – to combine applications from third-party providers into seamless, uninterrupted user experiences.

How SSO works

Single sign-on is based on a digital trust relationship between a group of related, trusted applications, web sites and services, called *service providers*, and an SSO solution, called an *identity provider*. The SSO solution is often part of a larger IAM (identity and access management) solution.

In general, SSO authentication works as follows:

1. A user logs into one of the trusted applications - or into a central portal connecting all the trusted applications (such as an employee portal or college student web site) - using SSO log in credentials.
2. When the user is successfully authenticated, the SSO solution generates a session authentication token containing specific information about the user's identity - a username, email address, etc. This token is stored with the user's web browser, or on the SSO or IAM server.

3. When the user attempts to access another of the trusted applications, the application checks with the SSO or IAM server to determine if user is already authenticated for the session. If so, the SSO solution validates the user by signing the authentication token with a digital certificate, and the user is granted access to the application. If not, the user is prompted to reenter log in credentials.

The process can vary depending on several factors. For example, a user who has been idle for a specified period may need to log in when they attempt to access another app. Or, if an authenticated user attempts an app or service that deals with particularly sensitive information, the user may be prompted for an additional authentication factor, such as a code sent to the user's mobile phone or email (see 'Adaptive SSO' below).

Benefits of SSO

Obviously, SSO saves users time and trouble. Take corporate users, for example: Instead of logging into multiple applications multiple times per day, with SSO they are often able be able to log into the corporate intranet or extranet just once for all-day access to every application they need.

But by dramatically reducing the number of passwords users need to remember and the number of user accounts administrators need to manage, SSO strengthens an organizations security posture. Specifically, SSO can

Replace password fatigue with one strong password. Users with lots of passwords to manage often lapse into using the same short, weak passwords - or slight variations thereof - for every application. A hacker who cracks one of these passwords can easily gain access to multiple applications. SSO can often reduce scores of short weak passwords to a single long, complex, strong password that's easier for users to remember - and much more difficult for hackers to break.
Help prevent unsafe password storage habits. SSO can reduce or eliminate the need for password managers, passwords stored in spreadsheets, passwords written on sticky notes and other memory aids - all of which make passwords easier for the wrong people to steal or stumble upon.
Reduce help desk calls - by a lot. According to industry analyst Gartner, 20 to 50 percent of IT help desk calls are related to forgotten passwords or password resets. Most SSO solutions make it easy for users to rest passwords themselves, with help desk assistance.
Give hackers a smaller target. According to IBM's Cost of a Data Breach 2021 report, compromised credentials were the most frequent initial attack vector for a data breach, accounting for 20% of all data breaches - and breaches that began with compromised credentials cost their victims $4.31 million on average. Fewer passwords mean fewer potential attack vectors.
Simplify management, provisioning and decommissioning of user accounts. With SSO, administrators have more centralized control over authentication requirements and access permissions. And when a user leaves the organization, administrators can remove permissions and decommission the user account in fewer steps.
Help simplify regulatory compliance. SSO meets or makes it easier to meet regulatory requirements around protection of personal identity information (PII) and data access control, as well as specific requirements in some regulations - such as HIPAA - around session time-outs.

SS0 risks

The chief risk of SSO is that if a user's credentials are compromised, they can grant an attacker access to all or most of the applications and resources on the network.

Requiring users to create long and complex passwords - and carefully encrypting and protecting them wherever they're stored - goes a long way toward preventing this worst-case scenario. But most security experts recommend implementing SSO with multi-factor authentication, or MFA. MFA requires users to provide at least one authentication factor in addition to a password - e.g., a code sent to a mobile phone, a fingerprint, or an ID card. Because these additional credentials are ones that hackers can't easily steal or spoof, MFA can dramatically reduce risks related to compromised credentials in SSO.

SSO variations

The SSO scheme describe above - a single log-in and set of user credentials providing session access to multiple related applications - is sometimes called simple or pure SSO. Other types of SSO - or authentication methods similar to SSO - include:

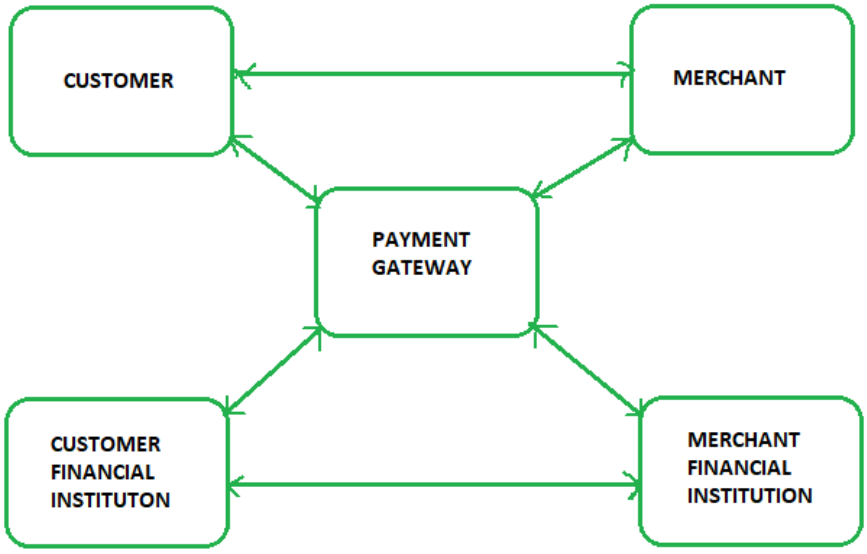
- **Adaptive SSO** initially requires a username and password at log-in, but subsequently requires additional authentication factors or a new log-in when additional risks emerge - such as when a user logs in from a new device or attempts to access particularly sensitive data or functionality.
- **Federated SSO** - more correctly called federated identity management (FIM) - is a superset of SSO. While SSO is based on a digital trust relationship among applications within a single organization's domain, FIM extends that relationship to trusted third parties, vendors, and other service providers outside the organization. For example, FIM might enable a logged-in employees to access third-party web applications, such as Slack or WebEx, without an additional log-in, or with a simple username-only log-in.
- **Social log-in** lets users use the same credentials they use to access popular social media sites to access third-party applications. Social log-in simplifies life for users. For third-party application providers, it can discourage undesirable behaviors (e.g., false logins, shopping cart abandonment) and provide valuable information for improving their apps.

12). SECURE INTER-BRANCH PAYMENT TRANSACTIONS

Secure Electronic Transaction or SET is a system that ensures the security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied to those payments. It uses different encryption and hashing techniques to secure payments over the internet done through credit cards. The SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT), and Netscape which provided the technology of Secure Socket Layer (SSL).

SET protocol restricts the revealing of credit card details to merchants thus keeping hackers and thieves at bay. The SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

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



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

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

Participants in SET :
In the general scenario of online transactions, SET includes similar participants:

1. Cardholder – customer
2. Issuer – customer financial institution
3. Merchant

4. Acquirer – Merchant financial
5. Certificate authority – Authority that follows certain standards and issues certificates(like X.509V3) to all other participants.

SET functionalities :

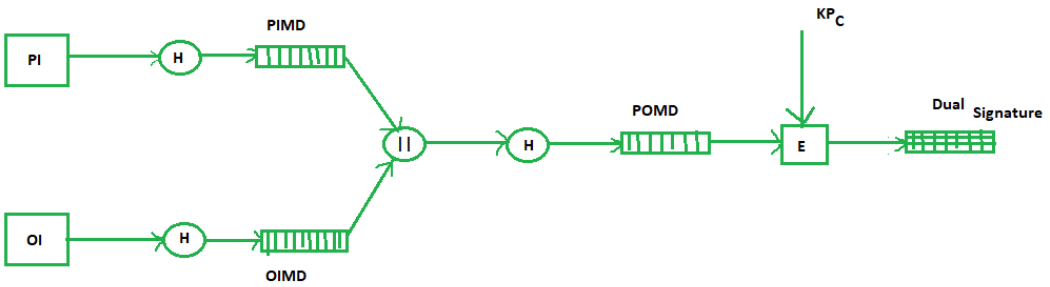
Provide Authentication

- Merchant Authentication – To prevent theft, SET allows customers to check previous relationships between merchants and financial institutions. Standard X.509V3 certificates are used for this verification.
- Customer / Cardholder Authentication – SET checks if the use of a credit card is done by an authorized user or not using X.509V3 certificates.

- Provide Message Confidentiality: Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purposes.
- Provide Message Integrity: SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1,

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

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



Where,

PI stands for payment information

OI stands for order information

PIMD stands for Payment Information Message Digest

OIMD stands for Order Information Message Digest

POMD stands for Payment Order Message Digest

H stands for Hashing

E stands for public key encryption

KPc is customer's private key

|| stands for append operation

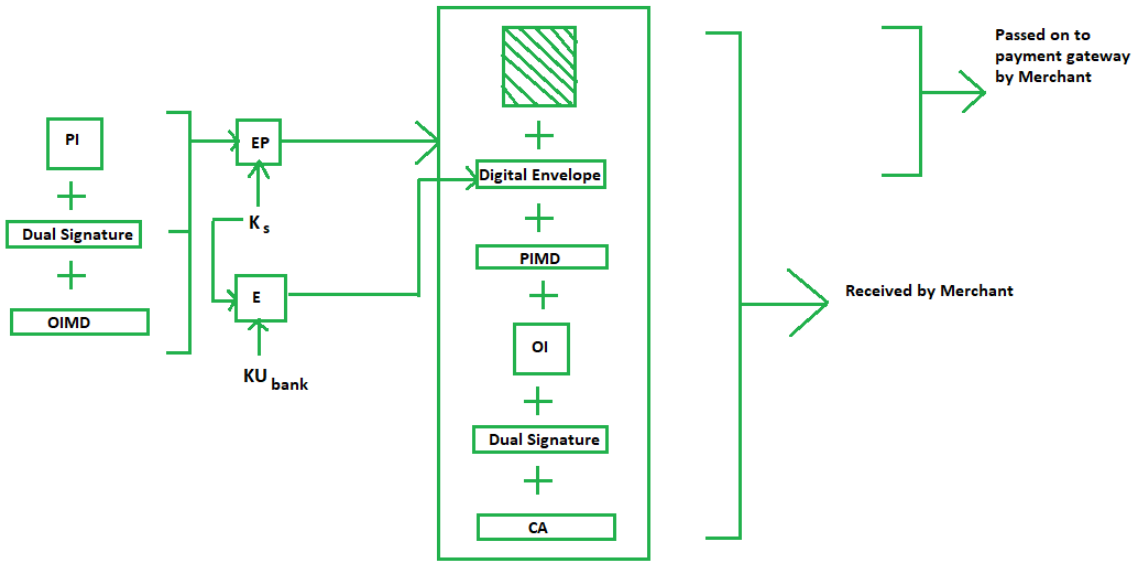
Dual signature, DS= E(KPc, [H(H(PI)||H(OI))])

Purchase Request Generation :

The process of purchase request generation requires three inputs:

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

The purchase request is generated as follows:



Here,

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

The new things are :

EP which is symmetric key encryption

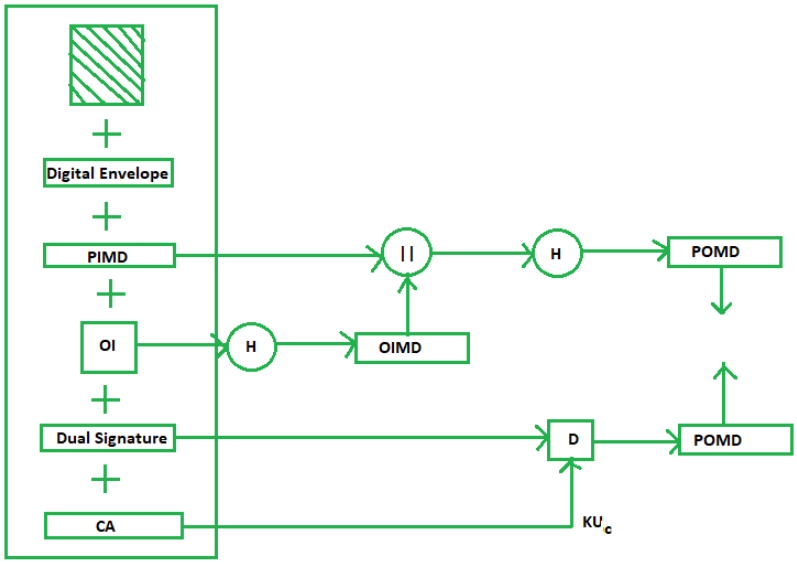
Ks is a temporary symmetric key

KUbank is public key of bank

CA is Cardholder or customer Certificate

Digital Envelope = E(KUbank, Ks)

Purchase Request Validation on Merchant Side :
The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows:

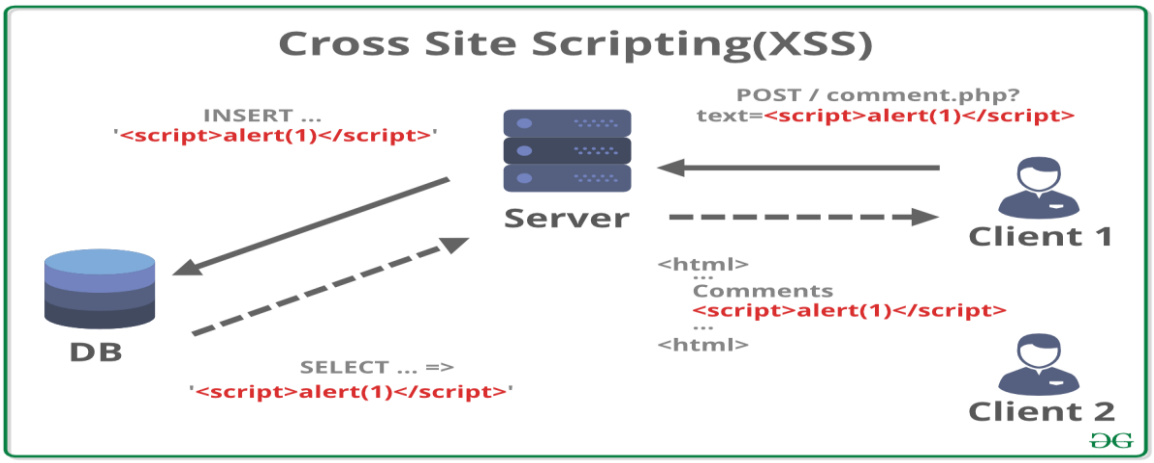


Since we used Customer's private key in encryption here we use KUC which is the public key of the customer or cardholder for decryption 'D'.

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

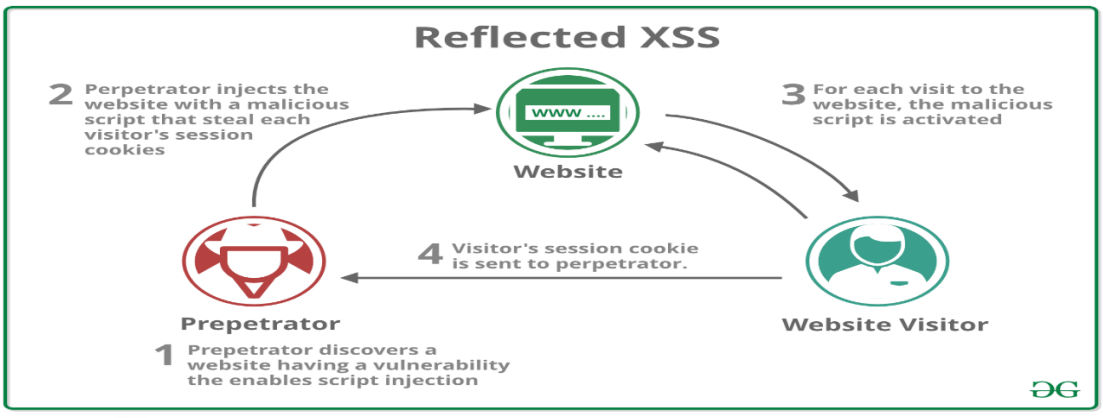
13). CROSS SITE SCRIPTING VULNERABILITY

Cross Site Scripting (XSS) is a vulnerability in a web application that allows a third party to execute a script in the user's browser on behalf of the web application. Cross-site Scripting is one of the most prevalent vulnerabilities present on the web today. The exploitation of XSS against a user can lead to various consequences such as account compromise, account deletion, privilege escalation, malware infection and many more.

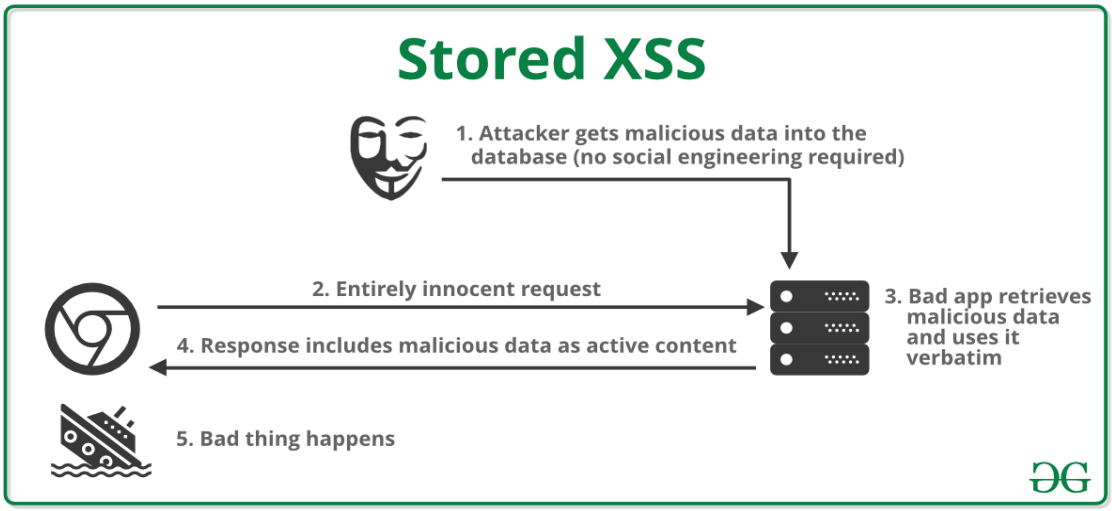


In its initial days, it was called CSS and it was not exactly what it is today. Initially, it was discovered that a malicious website could utilize JavaScript to read data from other website's responses by embedding them in an iframe, run scripts and modify page contents. It was called CSS (Cross Site Scripting) then. The definition changed when Netscape introduced the Same Origin Policy and cross-site scripting was restricted from enabling cross-origin response reading. Soon it was recommended to call this vulnerability as XSS to avoid confusion with Cascading Style Sheets(CSS). The possibility of getting XSSed arises when a website does not properly handle the input provided to it from a user before inserting it into the response. In such a case, a crafted input can be given that when embedded in the response acts as a JS code block and is executed by the browser. Depending on the context, there are *two types* of XSS –

Reflected XSS: If the input has to be provided each time to execute, such XSS is called reflected. These attacks are mostly carried out by delivering a payload directly to the victim. Victim requests a page with a request containing the payload and the payload comes embedded in the response as a script. An example of reflected XSS is XSS in the search field.



Stored XSS: When the response containing the payload is stored on the server in such a way that the script gets executed on every visit without submission of payload, then it is identified as stored XSS. An example of stored XSS is XSS in the comment thread.



There is another type of XSS called *DOM based XSS* and its instances are either reflected or stored. DOM-based XSS arises when user-supplied data is provided to the DOM objects without proper sanitizing. An example of code vulnerable to XSS is below, notice the variables *firstname* and *lastname* :

php

```
<?php


  if(isset($_GET["firstname"]) && isset($_GET["lastname"]))
  {


     $firstname = $_GET["firstname"];
     $lastname = $_GET["lastname"];


     if($firstname == "" or $lastname == "")
     {


        echo "<font color=\"red\">Please enter both fields...</font>";


     }
```

```
    else
    {

        echo "Welcome " . $firstname. " " . $lastname;

    }
}
?>
```

User-supplied input is directly added in the response without any sanity check. Attacker an input something like –

Html and it will be rendered as JavaScript. There are two aspects of XSS (and any security issue) –

<script> alert(1) </script>

1.Developer: If you are a developer, the focus would be secure development to avoid having any security holes in the product. You do not need to dive very deep into the exploitation aspect, just have to use tools and libraries while applying the best practices for secure code development as prescribed by security researchers. Some resources for developers are – a). OWASP Encoding Project : It is a library written in Java that is developed by the Open Web Application Security Project(OWASP). It is free, open source and easy to use. b). The "X-XSS-Protection" Header : This header instructs the browser to activate the inbuilt XSS auditor to identify and block any XSS attempts against the user. c). The XSS Protection Cheat Sheet by OWASP : This resource enlists rules to be followed during development with proper examples. The rules cover a large variety of cases where a developer can miss something that can lead to the website being vulnerable to XSS. d). Content Security Policy : It is a stand-alone solution for XSS like problems, it instructs the browser about "safe" sources apart from which no script should be executed from any origin.

2.Security researchers: Security researchers, on the other hand, would like similar resources to help them hunt down instances where the developer became lousy and left an entry point. Researchers can make use of – a). CheatSheets – 1. XSS filter evasion cheat sheet by OWASP. 2. XSS cheat sheet by Rodolfo Assis. 3. XSS cheat sheet by Veracode. b). Practice Labs – 1. bWAPP 2. DVWA(Damn vulnerable Web Application) 3. prompt.ml 4. CTFs c). Reports – 1. Hackerone Hactivity 2. Personal blogs of eminent security researchers like Jason Haddix, Geekboy, Prakhar Prasad, (Portswigger) etc.

INFORMATION SECURITY (CS4101PC)

Descriptive Questions:
a.2 Marks Questions

1.What is the need of public key ring and private key ring?

Public key ring is one of the data structures which is used to store the public keys of theother participants

Private Key ring is a data structure which is used to store the public and the private keys ofthe owner alone.

2.Mention the benefits of IPSec.

a.It provides strong security that can be applied to all traffic crossing the perimeter.

b.IPSec in a firewall is resistant to bypass.

c.IPSec is below the transport layer and so is transparent to applications.

d.IPSec is transparent to users.

3.List out the services provided by the IPSec.

a.Access control

b.Connectionless integrity

c.Data origin authentication

d.Rejection of replayed packets

e.Confidentiality

f.Limited traffic flow confidentiality

4.Name the protocols that provide security in IPSec.

i.Authentication header

ii.Encapsulating security payload

5.What is security association?

It is a one way relationship between a sender and a receiver that affords security services to the traffic carried on it.

6.Define transport and tunnel mode.

i.Transport mode provides protection primarily for upper layer protocols. Transport mode protection extends to the payload of an IP packet. Transport mode is used for end-to-end communication between two hosts.

ii.Tunnel mode provides protection to the entire packet. The entire packet (original packet) plus security fields is treated as the payload of new outer IP packet with a new outer IP header. Here the packet travels through a tunnel from one point of an IP network to another.

7.Why d o we need an anti replay service?

Anti replay service is required in order to avoid the duplicate packets (created by the opponent) which may cause disruption in the service.

8.What is the need pf padding in Encapsulating Security Payload (ESP)?

i.If an encryption algorithm requires the plaintext to be a multiple of some number of bytes,the padding field is used to expand the plaintext to the required length.

ii.ESP format requires that the pad length and the next header fields be right aligned within a32-bit word. The padding field is used to assure this alignment.

iii.Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

9.How the security associations be combined?

It can be done in two ways:

i.transport adjacency

ii.iterated tunneling

10.Define the terms: connection and session

i.A connection is a transport that provides a suitable type of service. Everyconnection is associated with one session.

ii.A session is an association between a client and a server. One session may havemultiple connections.

11.What is the use of the change cipher spec protocol?

This protocol consists of a single message which consists of a single byte with a value '1'. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite (cryptographic algorithm) to be used on the connection.

12.Mention the phases of the Handshake protocol.

a. Phase 1 – establish security capabilities
b. Phase 2 – server authentication and key exchange
c. Phase 3 – client authentication and key exchange
d. Phase 4 – finish

13.What is the need of Dual signature?

The purpose of the dual signature is to link two messages that are intended for twodifferent participants.

e.g., DS =EKR_C [H(H(PI) || H(OI))] where

DS –Dual signature

EKR_C – Encryption using the private key of the card holder (sender)

H – Hash function

PI – payment information

OI – order information

14.Classify the intruders.

i.Masquerader

ii.Misfeasor

iii.Clandestine user

15.How the password files be protected?

The password files can be protected in one of the two ways:

i.one way encryption ii.access control

16.Define firewall.

Firewall is the in which protects the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed.

17.What are the design goals of the firewall.

- All traffic from inside to outside, and vice versa, must pass through the firewall.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- It is immune to penetration.

18.List out the limitations of the firewall.

- It cannot protect against attacks that bypass the firewall.
- The firewall does not protect against internal threats.
- It cannot protect against the transfer of virus infected programs or files.

19.What are the types of firewall?

Packet filtering firewall

Application level gateway

Circuit level gateway

20.Define Bastion host.

A Bastion host is a system identified by the firewall administrator as a critical strong point in thenetwork security

21.List out the firewall configurations.

- Screened host firewall, single homed bastion
- Screened host firewall, dual homed bastion
- Screened subnet firewall

22.Define the two rules for multi-level security.

- No read up – a subject can only read on object of less or equal security level. This isreferred to as simple security property.
- No write down – a subject can only write into an object of greater or equal securitylevel. This is referred to as '*' property.

23.Define Trojan horse attack.

The Torjan horse attack begins with a hostile user, named X, gain legitimate access into the system and installs both the torjan horse program and a private file to be used in the attack as a 'back packet'.
X gives read / write permission to itself and gives Y
(authorized user) write-only permission. X now indicates Y to invoke torjan horse program, by advertising it as a useful utility. When the program detects that it is being executed by Y, it reads the sensitive character string from Y's file and copies it into X's back pocket file.

Define virus. Specify the types of viruses?

A virus is a program that can infect other program by modifying them the modification includes acopy of the virus program, which can then go on to infect other program,

Types:

1) Parasitic virus

2) Memory-resident virus

3) Boot sector virus

4) Polymorphic virus

5) Stealth virus

b.10 Marks Questions

1. A) Briefly explain about transport layer security and Padding.
   B) With a neat diagram, explain the operation of SSL and SSH Record Protocol.
2. A)List the five important features of IKE key determination algorithm
   B) What are the design goals for a firewall? Also mention its Limitations.
3. A) what are different Types of Viruses? Explain.
   B) Explain in detail about IP Security Policy.
4. A) Identify and describe different approaches of intrusion detection.
   B) What is the structure of virus?
5. A) Write short notes on non malicious program errors.
6. Write a short notes on : A) Intrusion detection b) Trusted System
7. A) Explain the various types of firewall.
   B) Explain the different firewall configuration.
8. A) Describe the various ways of combining the security associations.

B) What are the capabilities, limitations and design goals of firewalls?

9. A) Differentiate SSL & TLS

   B) Describe in detail, the payment processing of SET.

10. A) Explain in detail, the Handshake protocol in secure socket layer

    B) Explain the concept of password protection system