Department of Computer Science and Engineering

SUBJECT CODE: CS6701

SUBJECT NAME: Cryptography and network security

Regulation: 2013 **Semester and Year: 07/IV**

ANNA UNIVERSITY, CHENNAI-25

SYLLABUS COPY

REGULATION 2013

CS6701 CRYPTOGRAPHY AND NETWORK SECURITY L T P C 3 0 0 3

UNIT I INTRODUCTION & NUMBER THEORY 10

Services, Mechanisms and attacks-the OSI security architecture- Network security model- Classical Encryption techniques (Symmetric cipher model, substitution techniques, transposition techniques, steganography). FINITE FIELDS AND NUMBER THEORY: Groups, Rings, Fields-Modular arithmetic- Euclid's algorithm-Finite fields- Polynomial Arithmetic –Prime numbers-Fermat's and Euler's theorem- Testing for primality -The Chinese remainder theorem- Discrete logarithms.

UNIT II BLOCK CIPHERS & PUBLIC KEY CRYPTOGRAPHY 10

Data Encryption Standard-Block cipher principles-block cipher modes of operation-Advanced Encryption Standard (AES)-Triple DES-Blowfish-RC5 algorithm. **Public key cryptography:** Principles of public key cryptosystems-The RSA algorithm-Key management - Diffie Hellman Key exchange- Elliptic curve arithmetic-Elliptic curve cryptography.

UNIT III HASH FUNCTIONS AND DIGITAL SIGNATURES 8

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC –MD5 - SHA - HMAC – CMAC - Digital signature and authentication protocols – DSS – EI Gamal – Schnorr.

UNIT IV SECURITY PRACTICE & SYSTEM SECURITY 8

Authentication applications – Kerberos – X.509 Authentication services - Internet Firewalls for Trusted System: Roles of Firewalls – Firewall related terminology- Types of Firewalls - Firewall designs – SET for E-Commerce Transactions. Intruder – Intrusion detection system – Virus and related threats – Countermeasures – Firewalls design principles – Trusted systems – Practical implementation of cryptography and security.

UNIT V E-MAIL, IP & WEB SECURITY 9

E-mail Security: Security Services for E-mail-attacks possible through E-mail - establishing keys privacy-authentication of the source-Message Integrity-Non-repudiation-Pretty Good Privacy-S/MIME. **IPSecurity:** Overview of IPSec - IP and IPv6-Authentication Header-Encapsulation Security Payload (ESP)-Internet Key Exchange (Phases of IKE, ISAKMP/IKE Encoding). **Web Security:** SSL/TLS Basic Protocol-computing the keys- client authentication-PKI as deployed by SSLAttacks fixed in v3- Exportability-Encoding-Secure Electronic Transaction (SET).

TOTAL: 45 PERIODS

TABLE OF CONTENTS

AIM AND OBJECTIVE OF THE SUBJECT

The student should be made to

1. Understand OSI security architecture and classical encryption techniques.
2. Acquire fundamental knowledge on the concepts of finite fields and number theory.
3. Understand various block cipher and stream cipher models.
4. Describe the principles of public key cryptosystems, hash functions and digital signature.
5. Compare various Cryptographic Techniques
6. Design Secure applications
7. Inject secure coding in the developed applications

DETAILED LESSON PLAN

Text Book

1. William Stallings, Cryptography and Network Security, 6th Edition, Pearson Education, March 2013. (UNIT I, II, III, IV).

2. Charlie Kaufman, Radia Perlman and Mike Speciner, "Network Security", Prentice Hall of India, 2002. (UNIT V).

References

1. Behrouz A. Ferouzan, "Cryptography & Network Security", Tata McGraw Hill, 2007.

2. Man Young Rhee, "Internet Security: Cryptographic Principles", "Algorithms and Protocols", Wiley Publications, 2003.

3. Charles Pfleeger, "Security in Computing", 4th Edition, Prentice Hall of India, 2006.

4. Ulysess Black, "Internet Security Protocols", Pearson Education Asia, 2000.

5. Charlie Kaufman and Radia Perlman, Mike Speciner, "Network Security, Second Edition, Private Communication in Public World", PHI 2002.

6. Bruce Schneier and Neils Ferguson, "Practical Cryptography", First Edition, Wiley Dreamtech India Pvt Ltd, 2003.

7. Douglas R Simson "Cryptography – Theory and practice", First Edition, CRC Press, 1995.

8. http://nptel.ac.in/.

S.No	Unit No	Topic / Portions to be Covered	Hours Required / Planned	Cumulative Hrs	Books Referred
UNIT I INTRODUCTION & NUMBER THEORY					
1	I	Introduction	1	1	T1
2		Services, Mechanism, Attacks, OSI security architecture	1	2	T1
3		Network security model, Classical Encryption techniques - Symmetric cipher model	1	3	T1
4		Substitution techniques	1	4	T1
5		Transposition techniques, Steganography	1	5	T1
6		FINITE FIELDS AND NUMBER THEORY -Groups, Rings, Fields, Modular arithmetic	1	6	T1
7		Euclid's algorithm, Finite fields- Polynomial Arithmetic	1	7	T1
8		Prime numbers, Fermat's and Euler's theorem	1	8	T1
9		Testing for primality, The Chinese remainder theorem	1	9	T1
10		Discrete logarithms	1	10	T1
UNIT II BLOCK CIPHERS & PUBLIC KEY CRYPTOGRAPHY					
11	II	Data Encryption Standard, Block cipher principles	1	11	T1
12		Block cipher modes of operation	1	12	T1
13		Advanced Encryption Standard (AES)	1	13	T1
14		Triple DES	1	14	T1
15		Blowfish, RC5 algorithm	1	15	T1
16		Public key cryptography- Principles of public key cryptosystems	1	16	T1
17		The RSA algorithm	1	17	T1
18		Key management, Diffie Hellman Key exchange	1	18	T1
19		Elliptic curve arithmetic	1	19	T1

S.No	Unit No	Topic / Portions to be Covered	Hours Required / Planned	Cumulative Hrs	Books Referred
20		Elliptic curve cryptography	1	20	T1
UNIT III HASH FUNCTIONS AND DIGITAL SIGNATURES					
21	III	Authentication requirement, Authentication function	1	21	T1
22		MAC, Hash function	1	22	T1
23		Security of hash function and MAC	1	23	T1
24		MD5, SHA	1	24	T1
25		HMAC – CMAC	1	25	T1
26		Digital signature and authentication protocols	1	26	T1
27		DSS	1	27	T1
28		EI Gamal, Schnorr	1	28	T1
UNIT IV SECURITY PRACTICE & SYSTEM SECURITY					
29	IV	Authentication applications – Kerberos	1	29	T1
30		X.509 Authentication services	1	30	T1
31		Internet Firewalls for Trusted System: Roles of Firewalls, Firewall related terminology	1	31	T1
32		Types of Firewalls, Firewall designs	1	32	T1
33		SET for E-Commerce Transactions	1	33	T1
34		Intruder, Intrusion detection system	1	34	T1
35		Virus and related threats, Countermeasures	1	35	T1
36		Firewalls design principles, Trusted systems, Practical implementation of cryptography and security.	1	36	T1
UNIT V E-MAIL, IP & WEB SECURITY					
37		**E-mail Security:** Security Services for E-mail-attacks possible through E-mail,	1	37	T1

S.No	Unit No	Topic / Portions to be Covered	Hours Required / Planned	Cumulative Hrs	Books Referred
	V	Establishing keys privacy, Authentication of the source			
38		Message Integrity, Non-repudiation	1	38	T1
39		Pretty Good Privacy, S/MIME	1	39	T1
40		**IPSecurity:** Overview of IPSec - IP and IPv6	1	40	T1
41		Authentication Header, Encapsulation Security Payload (ESP)	1	41	T1
42		Internet Key Exchange (Phases of IKE, ISAKMP/IKE Encoding)	1	42	T1
43		**Web Security:** SSL/TLS Basic Protocol, computing the keys, client authentication	1	43	T1
44		PKI as deployed by SSL Attacks fixed in v3, Exportability, Encoding	1	44	T1
45		Secure Electronic Transaction (SET)	1	45	T1

UNIT I INTRODUCTION & NUMBER THEORY 10

Services, Mechanisms and attacks-the OSI security architecture-Network security model-Classical Encryption techniques (Symmetric cipher model, substitution techniques, transposition techniques, steganography).FINITE FIELDS AND NUMBER THEORY: Groups, Rings, Fields-Modular arithmetic-Euclid's algorithm-Finite fields- Polynomial Arithmetic –Prime numbers-Fermat's and Euler's theorem-Testing for primality -The Chinese remainder theorem- Discrete logarithms.

PART –A

1. **Differentiate between passive attacks and active attacks. (April/May 2015), (April/May'11), (May/June 2007).**

S.No	Passive Attack	Active Attack
1.	Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions.	Active Attacks involve some modification of the data stream or the creation of a false stream.
2.	Types: Release of message contents and traffic analysis	Types: Masquerade, replay, modification of message and denial of service.
3.	Very difficult to detect.	Easy to detect.
4.	The emphasis in dealing with passive attacks is on prevention rather than detection.	It is quite difficult to prevent active attacks absolutely.
5.	It does not affect the system.	It affects the system.

2. **What is the use of Fermat's theorem? (April/May 2011)**

Fermat's theorem sometimes is helpful for quickly finding a solution to some exponentiations and multiplicative inverses when the modulus is a prime.

3. **What is discrete logarithm? (April/May 2011), (May/June 2013)**

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

algorithm (DSA). The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number. If working with modulo arithmetic, and the base is a primitive root, then an integral discrete logarithm exists for any residue.

4. Why modular arithmetic has been used in cryptography? (Nov/Dec 2013)

Applications of modular are given to divisibility tests and to block ciphers in cryptography. Modular arithmetic directly underpins public key system such as RSA and Diffie-Hell man as well as providing finite fields which underlie elliptic curves and is used in a variety of symmetric key algorithms including AES, IDEA and RC4.

5. Find 11^7 mod 13. (April/May 2015)

To find 11^7 mod 13,

$11^2 = 121 \equiv 4 \pmod{13}$

$11^4 = (11^2)^2 \equiv 42 \equiv 3 \pmod{13}$

$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$

6. State Euler's theorem. (May/June 2014) (April/May 2010)

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

$$a^{\emptyset(n)} = 1 \; mod \; n$$

Proof: Let Z_n be the set of integers that are less than n and are relatively prime to n,

If a$\in$ Z_n we have

- (ax mod n) $\in$ Z_n
- If x , y $\in$ Z_n and x $\neq$ y then (ax mod p) $\neq$ (ay mod p)

So, $\prod_{x \in Zn} x$ = $\prod_{x \in Zn} (ax \; mod \; p)$ = (a^{p-1} mod p) $\prod_{x \in Zn} x$

7. Define finite field. (May/June 2012)

Finite field is a field that contains a finite number of elements. The finite fields are classified by size; there is exactly one finite field up to isomorphism of size p^k for each prime p and positive integer k.

8. What are the two basic functions used in encryption algorithms. (Nov/Dec 2014)

All the encryption algorithms are based on two general principles:

- **Substitution:** In which each element in the plaintext is mapped into another element.
- **Transposition:** In which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost.

9. Define threat and attack. (April/May 2010)

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

An attack on system security that derives from an intelligent threat: that is an intelligent act that is a deliberate attempt to evade security services and violate the security policy of a system.

10. Give the types of attack. (Nov/Dec 2011)

- Passive attack
- Active attacks

11. List out the problems of one time pad? (Nov/Dec 2011)

Problem with one time pad is that of making large quantities of random keys. It also makes the problem of key distribution and protection.

12. Define steganography. (May/June 2013)

Steganography is the art and science of writing hidden messages in such a way that no one , apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity.

13. What is the difference between a monoalphabetic cipher and a polyalphabetic cipher? (Nov/Dec 2012)

In monoalphabetic cipher single cipher alphabet is used per message. But in polyalphabetic cipher there are multiple ciphertext letters for each plaintext letter, one for each unique letter of keyword.

14. What is discrete logarithm problem? (May/June 2014)

Discrete Logarithm Problem(DLP) is easy to perform and hard to reverse. The strength of one way function is based on time needed to reverse it.

Let a cyclic finite group and g ∈ G be a generator of G. The DLP in G is following:

Given an element h ∈G , find the smallest positive integer x such that

h = [x]g (additive group)

$h = g^x$ (multiplicative group)

15. What do you mean by cryptanalysis? (**May/June 2012)**

Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. Cryptanalysis is what the layperson calls "breaking the code."The areas of cryptography and cryptanalysis together are called cryptology.

Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

PART-B

1. CLASSICAL ENCRYPTION TECHNIQUES- SUBSTITUTION AND TRANSPOSITION

- **Explain about substitution and transposition techniques with two examples for each. (May/June 2012) (April/May 2015)(April/May 2008) (April/May 2010) (Nov/Dec 2011)**
- **Write about any two classical crypto systems (substitution and transposition) with suitableexamples. (May/June 2013)(April/May 2011)**
- **Explain any two classical ciphers and also describe their security limitations. (May/June 2014)**

Substitution Techniques

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

Caesar Cipher

- The earliest known, and the simplest, use of a substitution cipher was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.
- For example, plain: meet me after the toga party
  cipher: PHHW PH DIWHU WKH WRJD SDUWB

- The alphabet is wrapped around, so that the letter following Z is A.
- The transformation are defined by listing all possibilities, as follows:

  plain: a b c d e f g h i j k l m n o p q r s t u v w x y z

  cipher: d e f g h i j k l m n o p q r s T u v w x y z a b c

A numerical equivalent to each letter:

a	B	c	d	E	f	g	h	i	j	K	l	m

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

n	O	p	q	R	s	t	u	v	w	X	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

- Then the algorithm can be expressed as follows. For each plaintext letter *p*, substitute the ciphertext letter *C*:

$$C = E(3, p) = (p + 3) \bmod 26$$

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

$$C = E(k, p) = (p + k) \bmod 26$$

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

$$p = D(k, C) = (C - k) \bmod 26$$

- If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: simply try all the 25 possible keys. The following figureshows the results of applying this strategy to the example ciphertext.
- Three important characteristics of this problem enabled us to use a bruteforce cryptanalysis:

1. The encryption and decryption algorithms are known.

2. There are only 25 keys to try.

3. The language of the plaintext is known and easily recognizable.

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

Figure: Brute force cryptanalysis of caesar cipher

Monoalphabetic Ciphers

- With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Before proceeding, the term *permutation can be defined*.
- A permutation of a finite set of elements *S* is an ordered sequence of all the elements of *S*, with each element appearing exactly once.
  For example, if *S* = {a, b, c}, there are six permutations of *S*:

  abc, acb, bac, bca, cab, cba

- In general, there are *n*! permutations of a set of *n* elements, because the first element can be chosen in one of *n* ways, the second in *n* - 1 ways, the third in *n* – 2 ways, and so on.

  plain: a b c d e f g h i j k l m n o p q r s t u v w x y z

  Caesar cipher: d e f g h i j k l m n o p q r s T u v w x y z a b c

- If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4 * 1026 possible keys.
- This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis. Such an approach is referred to as a monoalphabeticsubstitution cipher, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.
- Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.
- A countermeasure is to provide multiple substitutes known as homophones, for a single letter. For example, the letter e could be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone assigned to a letter in rotation or randomly.

Playfair Cipher

- The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertextdigrams.
- The Playfair algorithm is based on the use of a 5 * 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers's *Have His Carcase.*

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

- In this case, the keyword is *monarchy*. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.
- Plaintext is encrypted two letters at a time, according to the following rules:
  1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
  2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter tothe right, with the first element of the

row circularly following the last. For example, ar isencrypted as RM.

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

- The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 * 26 = 676 digrams, so that identification of individual digrams is more difficult. Furthermore, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis much more difficult.
- For these reasons, the Playfair cipher was for a long time considered unbreakable. It was used as the standard field system by the British Army in World War I and still enjoyed considerable use by the U.S. Army and other Allied forces during World War II.

Hill Cipher

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

The Hill Algorithm

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

$$c_1 = (k_{11}p_1 + k_{21}p_2 + k_{31}p_3) \bmod 26$$

$$c_2 = (k_{12}p_1 + k_{22}p_2 + k_{32}p_3) \bmod 26$$

$$c_3 = (k_{13}p_1 + k_{23}p_2 + k_{33}p_3) \bmod 26$$

This can be expressed in terms of row vectors and matrices:

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

or

$$\mathbf{C} = \mathbf{PK} \bmod 26$$

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

For example, consider the plaintext "paymoremoney" and use the encryption key

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

- The first three letters of the plaintext are represented by the vector (15 0 24). Then(15 0 24)**K** = (303 303 531) mod 26 = (17 17 11) = RRL. Continuing in this fashion, the ciphertext for the entire plaintext is RRLMWBKASPDH.
- Decryption requires using the inverse of the matrix K. We can compute det K = 23, and therefore, $(\det K)^{-1}$ mod 26 = 17. The inverse can be computed as

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} mod\ 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- It is easily seen that if the matrix $\mathbf{K}^{-1}$ is applied to the ciphertext, then the plaintext is recovered.
- In general terms, the Hill system can be expressed as

$$\mathbf{C} = E(\mathbf{K}, \mathbf{P}) = \mathbf{PK} \bmod 26$$

$$\mathbf{P} = D(\mathbf{K}, \mathbf{C}) = \mathbf{CK}^{-1} \bmod 26 = \mathbf{PKK}^{-1} = \mathbf{P}$$

- As with Playfair, the strength of the Hill cipher is that it completely hides single-letter frequencies. Indeed, with Hill, the use of a larger matrix hides more frequency information. Thus, a 3 * 3 Hill cipher hides not only single-letter but also two-letter frequency information.
- Although the Hill cipher is strong against a ciphertext-only attack, it is easily broken with a known plaintext attack.
- Consider this example. Suppose that the plaintext "hillcipher" is encrypted using a 2 * 2 Hill cipher to yield the ciphertext HCRZSSXNSP. Thus, (7 8)**K**mod 26 = (7 2); (11 11)**K** mod 26 = (17 25); and so on. Using the first two plaintext–ciphertext pairs,

$$\begin{bmatrix} 7 & 2 \\ 17 & 25 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 11 & 11 \end{bmatrix} K \bmod 26$$

The inverse of **X** can be computed:

$$\begin{bmatrix} 7 & 8 \\ 11 & 11 \end{bmatrix}^{-1} = \begin{bmatrix} 25 & 22 \\ 1 & 23 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{25} & \mathbf{22} \\ \mathbf{1} & \mathbf{23} \end{bmatrix} \begin{bmatrix} \mathbf{7} & \mathbf{2} \\ \mathbf{17} & \mathbf{25} \end{bmatrix} = \begin{bmatrix} \mathbf{549} & \mathbf{600} \\ \mathbf{398} & \mathbf{577} \end{bmatrix} \boldsymbol{mod\ 26} = \begin{bmatrix} \mathbf{3} & \mathbf{2} \\ \mathbf{8} & \mathbf{5} \end{bmatrix}$$

Polyalphabetic Ciphers

- Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message.
- The general name for this approach is polyalphabetic substitution cipher. All these techniques have the following features in common:
  1. A set of related monoalphabetic substitution rules is used.
  2. A key determines which particular rule is chosen for a given transformation.
- ***Vigenère Cipher***The best known, and one of the simplest, polyalphabetic ciphers is the Vigenère cipher. In this scheme, the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter, which is the ciphertext letter that substitutes for the plaintext letter a. Thus, a Caesar cipher with a shift of 3 is denoted by the key value 3.

- The Vigenère cipher can be expressed in the following manner. Assume a sequence of plaintext letters $P = p_0, p_1, p_2, \ldots, p_{n-1}$ and a key consisting of the sequence of letters $K = k_0, k_1, k_2, \ldots, k_{m-1}$, where typically $m < n$. The sequence of ciphertext letters $C = C_0, C_1, C_2, \ldots\ldots, C_{n-1}$ is calculated as follows:

  $C = C_0, C_1, C_2, \ldots, C_{n-1} = E(K, P) = E[(k_0, k_1, k_2, \ldots, k_{m-1}), (p_0, p_1, p_2, \ldots, p_{n-1})]$

  $= (p_0 + k_0) \bmod 26, (p_1 + k_1) \bmod 26, \ldots\ldots, (p_{m-1} + k_{m-1}) \bmod 26,$

  $(p_m + k_0) \bmod 26, (p_{m+1} + k_1) \bmod 26, \ldots\ldots, (p_{2m-1} + k_{m-1}) \bmod 26, \ldots$

- Thus, the first letter of the key is added to the first letter of the plaintext, mod 26, the second letters are added, and so on through the first m letters of the plaintext. For the next m letters of the plaintext, the key letters are repeated. This process continues until all of the plaintext sequence is encrypted. A general equation of the encryption process is

  $$C_i = (p_i + k_i \bmod m) \bmod 26$$

- A general equation for decryption is

  $$p_i = (C_i - k_i \bmod m) \bmod 26$$

- To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is *deceptive*, the message "we are discovered save yourself" is encrypted as

  Key :*deceptivedeceptivedeceptive*

  plaintext : wearediscoveredsaveyourself

  ciphertext : ZICVTWQNGRZGVTWAVZHCQYGLMGJ

- The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus, the letter frequency information is obscured. However, not all knowledge of the plaintext structure is lost.
- ***Vernam Cipher*** The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918.

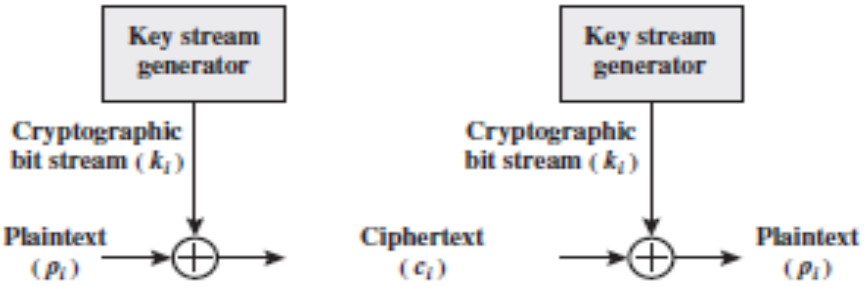


Figure: Vernam Cipher

- The system can be expressed as:

$$ci = pi \oplus ki$$

where

p_i= ith binary digit of plaintext

k_i= ith binary digit of key

c_i= ith binary digit of ciphertext

$\oplus$ = exclusive-or (XOR) operation

- Thus, the ciphertext is generated by performing the bitwise XOR of the plaintext and the key. Because of the properties of the XOR, decryption simply involves the same bitwise operation:

$$p_i = c_i \oplus k_i$$

One-Time Pad

- An Army Signal Corp officer, Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded.
- Each new message requires a new key of the same length as the new message. Such a scheme, known as a one-time pad, is unbreakable. It produces random output that bears no statistical relationship to the plaintext.
- Example:Consider a Vigenère scheme with 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message. Consider the ciphertext
ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

Two different decryptions are shown using two different keys:

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

key: *pxlmvmsydofuyrvzwctnlebnecvgdupahfzzlmnyih*

plaintext: mr mustard with the candlestick in the hall

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

key: *pftgpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt*

plaintext: miss scarlet with the knife in the library

- The security of the one-time pad is entirely due to the randomness of the key. If the stream of characters that constitute the key is truly random, then the stream of characters that constitute the ciphertext will be truly random. Thus, there are no patterns or regularities that a cryptanalyst can use to attack the ciphertext.
- The one-time pad offers complete security but, in practice, has two fundamental difficulties:
    1. There is the practical problem of making large quantities of random keys. Supplying truly random characters in this volume is a significant task.
    2. Even more daunting is the problem of key distribution and protection. For everymessage to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

- Because of these difficulties, the one-time pad is of limited utility and is useful primarily for low-bandwidth channels requiring very high security. The one-time pad is the only cryptosystem that exhibits what is referred to as *perfect secrecy*.

Transposition Techniques

- A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.
- The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.
- For example, to encipher the message “meet me after the toga party” with a rail fence of depth 2, we write the following:

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

e t e f e t e o a a t

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

- This sort of thing would be trivial to cryptanalyze. 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 the columns then becomes the key to the algorithm. For example,

  Key: 4 3 1 2 5 6 7

  Plaintext: a tt a c k p

  o s t p o n e

  d u n t i l t

  w o a m x y z

  Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

- The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed.

  Key: 4 3 1 2 5 6 7

  Input: t t n a a p t

  m t s u o a o

  d w c o i x k

  n l y p e t z

  Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

- To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is

  01 02 03 04 05 06 07 08 09 10 11 12 13 14

15 16 17 18 19 20 21 22 23 24 25 26 27 28

- After the first transposition,

  03 10 17 24 04 11 18 25 02 09 16 23 01 08

  15 22 05 12 19 26 06 13 20 27 07 14 21 28

  which has a somewhat regular structure. But after the second transposition,

  17 09 05 27 24 16 12 07 10 02 22 20 03 25

  15 13 04 23 19 14 11 01 26 21 18 08 06 28

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

2.TYPES OF ATTACKS

❖ **Explain briefly about the OSI Security Architecture.What are the different types of attacks? Explain. (Nov/Dec 2013)**

OSI Security Architecture

- **Threat**

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

- **Attack**

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt to evade security services and violate the security policy of a system.

- The OSI security architecture is useful to managers as a way of organizing the task of providing security. This architecture was developed as an international standard, computer and communications vendors have developed security features for their products and services that relate to this structured definition of services and mechanisms.

The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A process that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.
- **Security Attacks:**A useful means of classifying security attacks, used both in X.800 and RFC 4949, is in terms of *passive attacks* and *active attacks.* A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

➢ Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks arethe release ofmessage contents and traffic analysis

  - The **release of message contents:** A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. An opponent should be prevented from learning the contents of these transmissions.
  - **Traffic analysis:** Suppose that there is a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption.

➢ If encryption protection in place, an opponent might still be able to observe the pattern of these messages. This information might be useful in guessing the nature of the communication that was taking place.

➢ Passive attacks are very difficult to detect, because they do not involve any alteration of the data. Typically, the message traffic is sent and received in

an apparently normal fashion, and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.

- However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

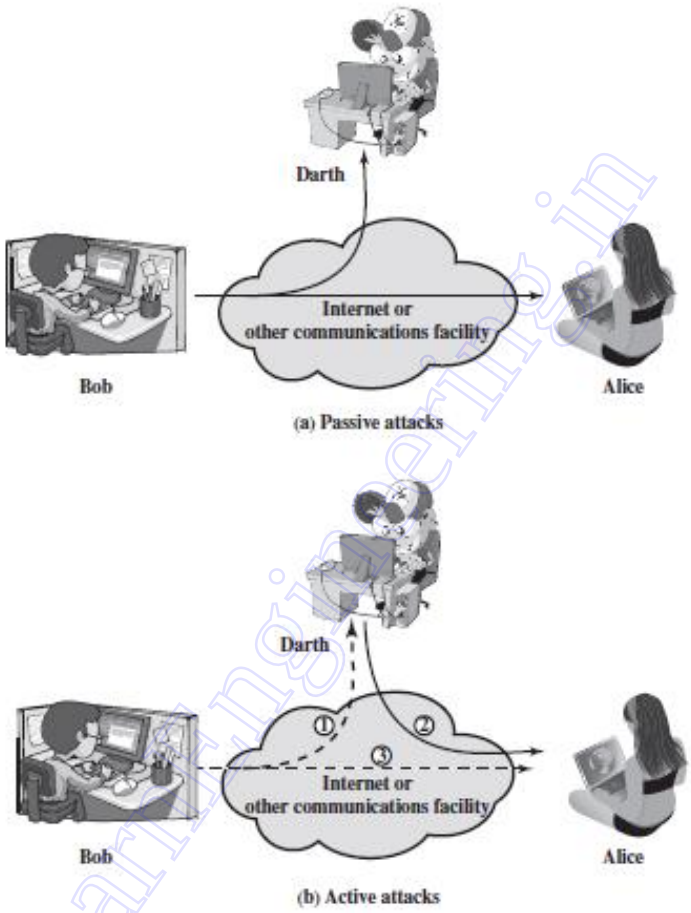


Figure: Active and Passive attacks

Active Attacks

- Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.
- A **masquerade** takes place when one entity pretends to be a different entity.
- **Replay** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- **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.

- The **denial of service** prevents or inhibits the normal use or management of communications facilities.
- Passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely

Security Services

- X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers. X.800 divides these services into five categories and fourteen specific services.

1. **Authentication**

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. Two specific authentication services are defined in X.800:

- **Peer entity authentication:** Provides for the corroboration of the identity of a peer entity in an association. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.
- **Data origin authentication:** Provides for the corroboration of the source of a data unit. It does not provide protection against the duplication or modification of data units.

2. **Access Control**

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

3. **Data Confidentiality**

Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of

protection can be identified. The broadest service protects all user data transmitted between two users over a period of time.

Table:Security Services (X.800)

AUTHENTICATION	**DATA INTEGRITY**
The assurance that the communicating entity is theone that it claims to be.	The assurance that data received are exactly assent by an authorized entity.
Peer Entity Authentication	**Connection Integrity with Recovery**
Used in association with a logical connection toprovide confidence in the identity of the entitiesconnected.	Provides for the integrity of all user data on aconnection and detects any modification, insertion,deletion, or replay of any data within an entire datasequence, with recovery attempted.
Data-Origin Authentication	**Connection Integrity without Recovery**
In a connectionless transfer, provides assurance that the source of received data is as claimed.	As above, but provides only detection without recovery.
ACCESS CONTROL	**Selective-Field Connection Integrity**
The prevention of unauthorized use of a resource.	Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.
DATA CONFIDENTIALITY	**Connectionless Integrity**
The protection of data from unauthorizeddisclosure.	Provides for the integrity of a single connectionlessdata block and may take the form of detection ofdata modification. Additionally, a limited form ofreplay detection may be provided.
Connection Confidentiality	**Selective-Field Connectionless**
The protection of all user data on a connection.	
Connectionless Confidentiality	
The protection of all user data in a single data block	
Selective-Field Confidentiality	
The confidentiality of selected fields within the user data on a connection or	

in a single data block. **Traffic-Flow Confidentiality** The protection of the information that might bederived from observation of traffic flows.	**Integrity** Provides for the integrity of selected fields within asingle connectionless data block. **NONREPUDIATION** Provides protection against denial by one of theentities involved in a communication of havingparticipated in all or part of the communication. **Nonrepudiation, Origin** Proof that the message was sent by the specified party. **Nonrepudiation, Destination** Proof that the message was received by the specifiedparty.

4. Data Integrity

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent with no duplication, insertion, modification, reordering, or replays. The connection-oriented integrity service addresses both message stream modification and denial of service.

On the other hand, a connectionless integrity service, one that deals with individual messages without regard to any larger context, generally provides protection against message modification only.

5. Nonrepudiation

Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

Security Mechanisms

The mechanisms are divided into those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol, and those that are not specific to any particular protocol layer or security service.

Table:Security Mechanisms (X.800)

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocollayer in order to provide some of the OSI securityservices. **Encipherment** The use of mathematical algorithms to transformdata into a form that is not readily intelligible. **Digital Signature** Data appended to, or a cryptographic transformationof, a data unit that allows a recipient of the data unitto prove the source and integrity of the data unit andprotect against forgery. **Access Control** A variety of mechanisms that enforce access rights toresources. **Data Integrity** A variety of mechanisms used to assure the integrityof a data unit or stream of data units. **Authentication Exchange** A mechanism intended to ensure the identity of anentity by means of	Mechanisms that are not specific to any particularOSI security service or protocol layer. **Trusted Functionality** That which is perceived to be correct with respect tosome criteria. **Security Label** The marking bound to a resource that names or designates the securityattributes of that resource. **Event Detection** Detection of security-relevant events. **Security Audit Trail** Data collected and potentially used to facilitate asecurity audit, which is an independent review andexamination of system records and activities. **Security Recovery** Deals with requests from mechanisms, such as eventhandling and management functions, and takesrecovery actions.

information exchange. **Traffic Padding** The insertion of bits into gaps in a data stream tofrustrate traffic analysis attempts. **Routing Control** Enables selection of particular physically secureroutes for certain data and allows routing changes, especially when a breach of security is suspected. **Notarization** The use of a trusted third party to assure certainproperties of a data exchange.	

X.800 distinguishes between reversible encipherment mechanisms and irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted. Irreversible encipherment mechanisms include hash algorithms and message authentication codes, which are used in digital signature and message authentication applications.

- The below table based on one in X.800, indicates the relationship between security services and security mechanisms.

	Mechanism							
Service	**Encipherment**	**Digital Signature**	**Access Control**	**Data Integrity**	**Authentication Exchange**	**Traffic Padding**	**Routing Control**	**Notarization**
Peer origin authentication	**Y**	**Y**			**Y**			

Data origin authentication	**Y**	**Y**						
Access control			**Y**					
Confidentiality	**Y**						**Y**	
Traffic flow confidentiality	**Y**					**Y**	**Y**	
Data integrity	**Y**	**Y**		**Y**				
Nonrepudiation		**Y**		**Y**				**Y**
Availability				**Y**	**Y**			

Table: Relationship Between Security Services and Mechanisms

3. FERMAT'S AND EULER'S THEOREM

❖ **Explain briefly about Fermat's and Euler's theorem. (April/May 2011) (Nov/Dec 2012)(Nov/Dec 2014)(Nov/Dec 2013)(April/May 2015)**

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

Fermat's Theorem

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then

$$a^{p-1} \equiv 1 \pmod{p}$$

Proof: Consider the set of positive integers less than p: {1, 2, c, p - 1} and multiply each element by a, modulo p, to get the set X = {a mod p, 2a mod p, c, (p - 1)$a$mod p}. None of the elements of X is equal to zero because p does not divide a. Furthermore, no two of the integers in X are equal.

- To see this, assume that $ja \equiv ka \pmod{p}$, where $1 \leq j < k \leq p - 1$. Because a is relatively prime to p, Eliminate a from both sides of the equation resulting in $j \equiv k \pmod{p}$.
- This last equality is impossible, because j and k are both positive integers less than p. Therefore, $(p - 1)$ elements of X are all positive integers with no two elements equal.
- We can conclude the X consists of the set of integers $\{1, 2, \ldots, p - 1\}$ in some order. Multiplying the numbers in both sets (p and X) and taking the result mod p yields

$$a \times 2a \times \ldots\ldots\ldots \times (p - 1)a \equiv [(1 \times 2 \times \ldots\ldots\ldots \times (p - 1)] \pmod{p}$$

$$a^{p-1}(p - 1)! \equiv (p - 1)! \pmod{p}$$

- We can cancel the $(p - 1)!$ term because it is relatively prime to p. This yields Equation, which completes the proof.

$$a = 7, p = 19$$

$$7^2 = 49 \equiv 11 \pmod{19}$$

$$7^4 \equiv 121 \equiv 7 \pmod{19}$$

$$7^8 \equiv 49 \equiv 11 \pmod{19}$$

$$7^{16} \equiv 121 \equiv 7 \pmod{19}$$

$$a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}$$

- An alternative form of Fermat's theorem is also useful: If p is prime and a is a positive integer, then

$$a^p \equiv a \pmod{p}$$

Euler's Totient Function

Euler's totient function, $f(n)$, is defined as the number of positive integers less than n and relatively prime to n. By convention, $\Phi(1) = 1$.

Determine Φ (37) and Φ (35).

Because 37 is prime, all of the positive integers from 1 through 36 are relatively

prime to 37. Thus Φ (37) = 36.

To determine Φ (35), we list all of the positive integers less than 35 that are relatively

prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18

19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

There are 24 numbers on the list, so Φ (35) = 24.

- The below table lists the first 30 values of f(n). The value f(1) is without meaning but is defined to have the value 1.
- It should be clear that, for a prime number p,

$$\Phi(p) = p - 1$$

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

$$\Phi(n) = \Phi(pq) = \Phi(p) \times \Phi(q) = (p - 1) \times (q - 1)$$

  - To see that $\Phi(n) = \Phi(p) \times \Phi(q)$, consider that the set of positive integers less that n is the set $\{1,\ldots\ldots, (pq - 1)\}$. The integers in this set that are not relatively prime to n are the set $\{p, 2p,\ldots.., (q - 1)p\}$ and the set $\{q, 2q,\ldots\ldots, (p - 1)q\}$. Accordingly,

$$\Phi(n) = (pq - 1) - [(q - 1) + (p - 1)]$$

$$= pq - (p + q) + 1$$

$$= (p - 1) \times (q - 1)$$

$$= \Phi(p) \times \Phi(q)$$

Table: Some Values of Euler's Totient Function $\phi(n)$

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

$$\phi(21) = \phi(3) \times \phi(7) = (3-1) \times (7-1) = 2 \times 6 = 12$$

Where the 12 integers are {1,2,4,5,8,10,11,13,16,17,19,20}

Euler's Theorem

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

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

Proof: The above equation is true if *n* is prime, because in that case, $\Phi(n) = (n - 1)$ and Fermat's theorem holds. However, it also holds for any integer *n*. $\Phi(n)$ is the number of positive integers less than *n* that are relatively primeto *n*.

Consider the set of such integers, labeled as

$$R = \{x_1, x_2, \ldots\ldots, x_{\Phi(n)}\}$$

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

Now multiply each element by a, modulo n:

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

- The set S is a permutation of R, by the following line of reasoning:

  Because a is relatively prime to n and x_i is relatively prime to n, ax_i must also be relatively prime to n. Thus, all the members of S are integers that are less than n and that are relatively prime to n.

If $ax_i \bmod n = ax_j \bmod n$, then $x_i = x_j$.

Therefore,

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

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

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

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

which completes the proof. This is the same line of reasoning applied to the proof of Fermat's theorem.

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

As is the case for Fermat's theorem, an alternative form of the theorem is also useful:

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

4. CHINESE REMINDER THEOREM

❖ **Describe Chinese Reminder theorem. (April/May 2011), (May/June 2012)(Nov/Dec 2013), (Nov/Dec 2014)**

One of the most useful results of number theory is the **Chinese remainder theorem**(CRT). In essence, the CRT says it is possible to reconstruct integers in a certainrange from their residues modulo a set of pairwise relatively prime moduli.

The CRT can be stated in several ways. Let

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

where the $m_i$are pairwise relatively prime; that is, gcd(m_i, m_j) = 1 for 1 ≤i, j ≤k, and $i \neq j$. Any integer A in $Z_M$can be represented by a k-tuple whose elements are in Z_{mi}using the following correspondence:

$$A \leftrightarrow (a_1, a_2, \ldots, a_k)$$

where$A \in Z_{M,}ai \in Z_{mi}$, and $a_i = A \bmod m_i$for 1 ≤i ≤k. The CRT makes two assertions.The mapping of the above equation is a one-to-one correspondence (called a **bijection**) between $Z_M$and the Cartesian product $Z_{m1} \times Z_{m2} \times \ldots\ldots \times Z_{mk}$. That is, for every integer A such that 0 ≤A ≤M, there is a unique k-tuple (a_1, a_2,…….., a_k) with 0 ≤a_i<$m_i$that represents it, and for every such k-tuple (a_1, a_2,……, a_k), there is a unique integer A in Z_M.

- **First assertion:** The transformation from A to(a_1, a_2,…., a_k), is obviously unique; that is, each $a_i$is uniquely calculated as $a_i = A \bmod m_i$. Computing A from (a_1, a_2,….., a_k) can be done as follows.

Let $M_i = M/mi$ for 1 ≤i ≤k. Note that $Mi = m_1 \times m_2 \times \ldots\ldots \times m_{i-1} \times m_{i+1} \times \ldots.. \times m_k$, so that $M_i \equiv 0 \pmod{m_j}$ for all $j \neq i$. Then let

$$c_i = M_i \times (M_i^{-1} mod\ m_i) \qquad for\ 1 \leq i \leq k$$

By the definition of M_i, it is relatively prime to $m_i$and therefore has a unique multiplicativeinverse mod m_i. Now compute

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i \right) (mod\ M)$$

There is a need to show that $a_i = A \bmod m_i$for 1 ≤i ≤k. Note that $c_j \equiv Mj \equiv 0 \pmod{m_i}$ if $j \neq i$, and that $c_i \equiv 1 \pmod{m_i}$. Itfollows that $a_i = A \bmod m_i$.

- The **second assertion** of the CRT, concerning arithmetic operations, follows from the rules for modular arithmetic. That is, the second assertion can be stated as follows: If

$$A \leftrightarrow (a_1, a_2, \ldots, a_k)$$
$$B \leftrightarrow (b_1, b_2, \ldots, b_k)$$

Then

$$(A + B) mod\ M \leftrightarrow \left((a_1 + b_1) mod\ m_1, \ldots, (a_k + b_k) mod\ m_k\right)$$
$$(A - B) mod\ M \leftrightarrow \left((a_1 - b_1) mod\ m_1, \ldots, (a_k - b_k) mod\ m_k\right)$$
$$(A \times B) mod\ M \leftrightarrow \left((a_1 \times b_1) mod\ m_1, \ldots, (a_k \times b_k) mod\ m_k\right)$$

One of the useful features of the Chinese remainder theorem is that it provides a way to manipulate (potentially very large) numbers mod M in terms of tuples of smaller numbers. This can be useful when M is 150 digits or more.

The Euclidean Algorithm

One of the basic techniques of number theory is the Euclidean algorithm, which is a simple procedure for determining the greatest common divisor of two positive integers. Two integers are **relatively prime**if their only common positive integer factor is 1.

Greatest Common Divisor

Nonzero *b* is defined to be a divisor of *a*if *a* = *mb*for some *m*, where *a*, *b*, and *m* are integers. The notation gcd(*a*, *b*) will be used to mean the **greatest common divisor** of *a* and *b*. The greatest common divisor of *a* and *b* is the largest integer that divide both *a* and *b*,definegcd(0, 0) = 0.

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

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

An equivalent definition is the following:

gcd(*a*, *b*) = max[*k*, such that *k* | *a* and *k*| *b*]

The greatest common divisor should be positive,

gcd(*a*, *b*) =gcd(*a*, -*b*) = gcd(-*a*, *b*) = gcd(-*a*,-*b*).

In general, gcd(*a*, *b*) = gcd(| *a*| , | *b*|).

Eg.gcd(60,24)=gcd(60,-24)=12

Also, because all nonzero integers divide 0, gcd(a, 0) = $|a|$.

Two integers a and$b$ are relatively prime if their only common positive integer factor is 1. This is equivalent to saying that a and b are relatively prime if gcd(a, b) = 1.

Finding the Greatest Common Divisor

An algorithm can be credited to Euclid for easily finding the greatest common divisor of two integers. Suppose we have integers a, b such that d = gcd(a, b). Because gcd($|a|$, $|b|$) = gcd(a, b), there is no harm in assuming $a \geq b > 0$. Now dividing a by b and applying the division algorithm, we can state:

$$a = q_1 b + r_1 \quad 0 \leq r_1 < b$$

If it happens that $r_1 = 0$, then $b|a$and d = gcd(a, b) = b. But if $r_1 \neq 0$, we can state that $d|r_1$. This is due to the basic properties of divisibility: the relations $d|a$ and $d|b$ together imply that $d|(a - q_1 b)$, which is the same as $d|r_1$. Since$b > r_1$, we can divide b by $r1$and apply the division algorithm to obtain:

$$b = q_2 r_1 + r_2 \qquad 0 \leq r_2 < r_1$$

If $r_2 = 0$, then $d = r_1$ and if $r_2 \neq 0$, then d = gcd(r_1, r_2). The division process continues until some zero remainder appears, say, at the (n + 1)th stage where r_{n-1} is divided by r_n. The result is the following system of equations:

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

At each iteration, d = gcd(r_i, r_{i+1}) until finally d = gcd(r_n, 0) = r_n.

Thus, the greatest common divisor of two integers can be found by repetitive application of the division algorithm. This scheme is known as the Euclidean algorithm.

- The first step is to show that r_n divides a and$b$. It follows from the last division in above equation that $r_n$divides r_{n-1}.The next to last division shows that $r_n$divides r_{n-2} because it divides both terms on the right.

- Successively, one sees that $r_n$divides all r_i's and finally *a* and *b*. It remains to show that $r_n$is the largest divisor that divides *a* and*b*.
- *c* must divide r_n, so that r_n= gcd(*a*, *b*).
- An example with relatively large numbers to see the power of this algorithm:

To find d=gcd(a,b) = gcd(1160718174,316258250)		
$a=q_1b+r_1$	1160718174= 3 ×316258250+ 211943424	d=gcd(316258250,211943424)
$b=q_2r_1+r_2$	316258250= 1 ×211943424+ 104314826	d=gcd(211943424, 104314826)
$r_1=q_3r_2+r_3$	211943424= 2 ×104314826+ 3313772	d=gcd(104314826, 3313772)
$r_2=q_4r_3+r_4$	104314826= 31 ×3313772+ 1587894	d=gcd(3313772, 1587894)
$r_3=q_5r_4+r_5$	3313772= 2 ×1587894+ 137984	d=gcd(1587894, 137984)
$r_4=q_6r_5+r_6$	1587894= 11 ×137984+ 70070	d=gcd(137984,70070)
$r_5=q_7r_6+r_7$	137984= 1 ×70070+ 67914	d=gcd(70070, 67914)
$r_6=q_8r_7+r_8$	70070= 1 ×67914+ 2156	d=gcd(67914, 2156)
$r_7=q_9r_8+r_9$	67914= 31 ×2156+ 1078	d=gcd(2156, 1078)
$r_8=q_{10}r_9+r_{10}$	2156= 2 ×1078+ 0	d=gcd(1078,0) = 1078
Therefore, gcd(1160718174,316258250) = 1078		

Dividend	**Divisor**	**Quotient**	**Remainder**
a=1160718174	b=316258250	q_1=3	r_1=211943424
b=316258250	r_1=211943424	q_2=1	r_2=104314826
r_1=211943424	r_2=104314826	q_3=2	r_3=3313772
r_2=104314826	r_3=3313772	q_4=31	r_4=1587894
r_3=3313772	r_4=1587894	q_5=2	r_5=137984
r_4=1587894	r_5=137984	q_6=11	r_6=70070
r_5=137984	r_6=70070	q_7=1	r_7=67914
r_6=70070	r_7=67914	q_8=1	r_8=2156

r_7=67914	r_8=2156	q_9=31	r_9=1078
r_8=2156	r_9=1078	q_{10}=2	r_{10}=0

Table :Euclidean Algorithm Example

- In this example, we begin by dividing 1160718174 by 316258250, which gives 3 with a remainder of 211943424. Next we take 316258250 and divide it by 211943424. The process continues until we get a remainder of 0, yielding a result of 1078.
- For every step of the iteration, we have $r_{i-2} = q_i r_{i-1} + r_i$, where r_{i-2} is the dividend, r_{i-1} is the divisor, $q_i$is the quotient, and $r_i$is the remainder.

5. MODULAR ARITHMETIC

❖ **Explain the Modular Arithmetic operation and properties indetail. (R-13)**

The Modulus

- If a is an integer and n is a positive integer, we define a mod n to be the remainder when a is divided by n. The integer n is called the **modulus**. Thus, for any integer a, we can write follows:

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

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

11 mod 7 = 4;	-11 mod 7 = 3

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

73 = 4 (mod 23);	21 = 19 (mod 10)

if$a \equiv 0 \pmod n$, then $n|a$.

Properties of Congruences

Congruences have the following properties:

1. $a \equiv b \pmod n$ if $n|(a - b)$.

2. $a \equiv b \pmod n$ implies $b \equiv a \pmod n$.

3. $a \equiv b \pmod n$ and $b \equiv c \pmod n$ imply $a \equiv c \pmod n$.

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

23 = 8 (mod5) because 23 – 8 = 15 = 5×3
-11 = 5 (mod 8) because -11-5 = -16 = 8 × (-2)
81 = 0 (mod 27) because 81 – 0 = 81 = 27 × 3

Modular Arithmetic Operations

- The (mod n) operator maps all integers into the set of integers {0, 1, c, (n - 1)}. This technique is known as **modular arithmetic**.
  Modular arithmetic exhibits the following properties:

1. [(a mod n) + (b mod n)] mod n = (a + b) mod n

2. [(a mod n) - (b mod n)] mod n = (a - b) mod n

3. [(a mod n) × (b mod n)] mod n = (a ×b) mod n

First property:

- Define (a mod n) = $r_a$and (b mod n) = r_b. Then we can write $a = r_a + jn$for some integer j and $b = r_b + kn$for some integer k.
  Then

$$(a + b) \bmod n = (ra + jn + rb + kn) \bmod n = (ra + rb + (k + j)n) \bmod n$$
$$= (ra + rb) \bmod n$$
$$= [(a \bmod n) + (b \bmod n)] \bmod n$$

  - Examples of the three properties:

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

  - Exponentiation is performed by repeated multiplication, as in ordinary arithmetic.

To find 11^7 mod 13,

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

- Thus, the rules for ordinary arithmetic involving addition, subtraction, and multiplication carry over into modular arithmetic. The following table below provides an illustration of modular addition and multiplication modulo 8.

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

(a) Addition modulo 8

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

(b) Multiplication modulo 8

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

(c) Additive and multiplicative inverse modulo 8

Table: Arithmetic Modulo 8

- Both matrices are symmetric about the main diagonal in conformance to the commutative property of addition and multiplication.
- As in ordinary addition, there is an additive inverse, or negative, to each integer in modular arithmetic.
- In this case, the negative of an integer x is the integer y such that $(x + y) \text{ mod } 8 = 0$.
- To find the additive inverse of an integer in the left-hand column, scan across the corresponding row of the matrix to find the value 0; the integer at the top of that column is the additive inverse; thus, $(2 + 6) \text{ mod } 8 = 0$.Similarly, the entries in the multiplication table are straightforward.
- In modular arithmetic mod 8, the multiplicative inverse of x is the integer y such that $(x \times y) \text{ mod } 8 = 1 \text{ mod } 8$.

Properties of Modular Arithmetic

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

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

- This is referred to as the **set of residues**, or **residue classes** (mod n). To be more precise, each integer in Zn represents a residue class. We can label the residue classes (mod n) as [0], [1], [2], ….., [n - 1], where

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

The residue classes (mod 4) are
[0] = {…,-16,-12,-8,-4,0,4,8,12,16,…}
[1] = {…,-15,-11,-7,-3,1,5,9,13,17,…}
[2] = {…,-14,-10,-6,-2,2,6,10,14,18,…}
[3] = {…,-13,-9,-5,-1,3,7,11,15,19,…}

- Of all the integers in a residue class, the smallest nonnegative integer is the one used to represent the residue class.
- Finding the smallest nonnegative integer to which k is congruent modulo n is called **reducing *k* modulo *n***.
- Z_n is a commutative ring with a multiplicative identity element.

Property	Expression
Commutative Laws	(w+x)mod n – (x+w) mod n) (w×x) mod n – (x×w) mod n)
Associative Laws	[(w+x) +y]mod n – [w+(x+y)] mod n [(w×x) ×y] mod n – [w×(x×y)] mod n
Distributive Laws	[w×(x+y)] mod n – [(w×x)+(w×y)] mod n
Identities	(0+w) mod n = w mod n (1 × w) mod n = w mod n
Additive Inverse (-w)	For each w ∈ Z_n, there exists a z such that w+z=0 mod n

- There is one peculiarity of modular arithmetic that sets it apart from ordinary arithmetic.

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

$(5 + 23) = (5 + 7) \pmod 8$; $23 = 7 \pmod 8$

- Equation above is consistent with the existence of an additive inverse. Adding the additive inverse of a to both sides of Equation above, we have

$$((-a) + a + b) \equiv ((-a) + a + c) \pmod n$$

$$b \equiv c \pmod n$$

- However, the following statement is true only with the attached condition:

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

- Two integers are **relatively prime** if their only common positive integer factor is 1.
- Applying the multiplicative inverse of a to both sides, we have

$$((a^{-1})ab) \equiv ((a^{-1})ac) \pmod n$$

$$b \equiv c \pmod n$$

- In general, an integer has a multiplicative inverse in Z_n if that integer is relatively prime to n.

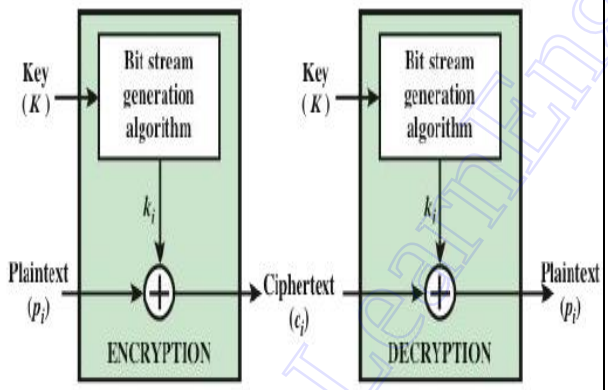
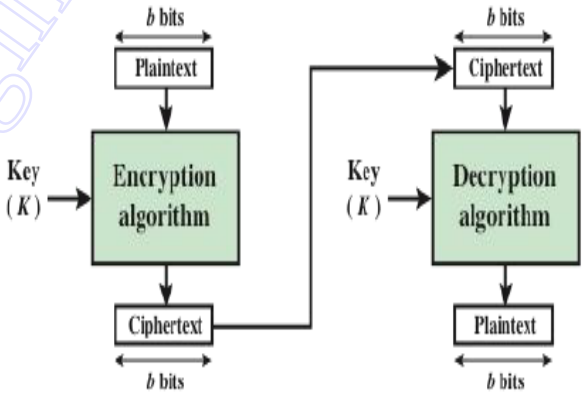
UNIT II BLOCK CIPHERS & PUBLIC KEY CRYPTOGRAPHY 10

Data Encryption Standard-Block cipher principles-block cipher modes of operation-Advanced Encryption Standard (AES)-Triple DES-Blowfish-RC5 algorithm. Public key cryptography: Principles of public key cryptosystems-The RSA algorithm-Key

management - Diffie Hellman Key exchange- Elliptic curve arithmetic-Elliptic curve cryptography.

PART-A

1. What is difference between a block cipher and a stream cipher?
(MAY/JUNE 2012), (APR/MAY 2015)

Stream Cipher	Block Cipher
1. A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyedVigenère cipherand the Vernam cipher.	1. 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 length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key. It has broader range of applications.
 (a) Stream Cipher Using Algorithmic Bit Stream Generator	 (b) Block Cipher

2. What is key distribution center? (MAY/JUNE 2012)

For symmetric key cryptography, the trusted intermediaryis called a **Key Distribution Center (KDC**), which is a single, trusted network entity with whom one

has established ashared secret key. A key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed. Each user must share a unique key with the key distribution center for purposes of key distribution. The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.

3. Mention the application of public key cryptography. (MAY/JUNE 2012)

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

4. State whether symmetric and asymmetric cryptographic algorithms require key exchange. (MAY/JUNE 2014)

Both symmetric and asymmetric cryptographic algorithms require key exchange. Key exchange (also known as "key establishment") is any method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. If the cipher is a symmetric key cipher, both will need a copy of the same key. If an asymmetric key cipher with the public/private key property, both will need the other's public key.

5. Write down the difference between the public key and private key cryptosystems. (MAY/JUNE 2012)

Public key cryptosystems	Private key cryptosystems
1. One algorithm is used for encryption and decryption with pair of keys.	1. Same algorithm and same key is used for encryption and decryption.
2. The sender and receiver must each have one of the matched pair of keys. One	2. Sender and receiver must share the algorithm and key. Key must be kept

of two keys must be kept secret.	secret.

6. Is it possible to use the DES algorithm to generate message authentication code? Justify. (NOV/DEC 2014)

Yes. It can use any block cipher chaining mode and use final block as a MAC. Data Authentication Algorithm(DAA) is a widely used MAC based on DES-CBC. Encrypt message using CBC mode and send just the final block as the MAC.

7. What is triple DES? (APR/MAY2010)

Triple DES involves repeating the DES algorithm three times on the plaintext using two or three different keys to produce the ciphertext.

8. Write down the purpose of the S-Boxes in DES. (NOV/DEC 2011)

S-Box is a nonlinear, invertible matrix in which each row defines a general reversible substitution. DES consists of 8 S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.

9. Define : Diffusion. (NOV/DEC 2011)

Diffusion is one of the basic building block for any cryptographic system. In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits. This is possible by using permutation.

10. List out the parameters of AES. (NOV/DEC 2011)

- Key Size (words/bytes/bits)
- Plaintext Block Size (words/bytes/bits)
- Number of Rounds
- Round Key Size (words/bytes/bits)
- Expanded Key Size (words/bytes)

11. State the difference between conventional encryption and public-key encryption. (NOV/DEC 2011)

Conventional encryption	Public-key encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key.	Needed to Work: 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	Needed for Security: 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

12. State few applications of RC4 algorithm. (APR/MAY 2015)

RC4 is used in SSL/TLS. It is also used in WEP, the IEEE 802.11 wireless networking security standard. It can also be found in a number of other applications including email encryption products.

13. Define primitive root. (NOV/DEC 2012)

A primitive root of a prime number p is one whose powers modulo p generate all the integers from 1 to p - 1. That is, if a is a primitive root of the prime number p, then the numbers

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

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

PART-B

1. DATA ENCRYPTION STANDARD (DES)

- **Explain about the single round of DES algorithm and the key discarding process of DES.(16) (APR/MAY 2011)**
- **Describe the working principle of simple DES with an example. (16) (MAY/JUNE 2014), (APR/MAY 2015), (MAY/JUNE 2013) (NOV/DEC 2012)**
- **Explain the key generation, encryption and decryption of SDES algorithm. (16) (NOV/DEC 2011), (NOV/DEC 2014)**

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA).

For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

DES Encryption

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

At the left hand side of the figure, the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that

rearranges the bits to produce the *permuted input*. This is followed by a phase consisting

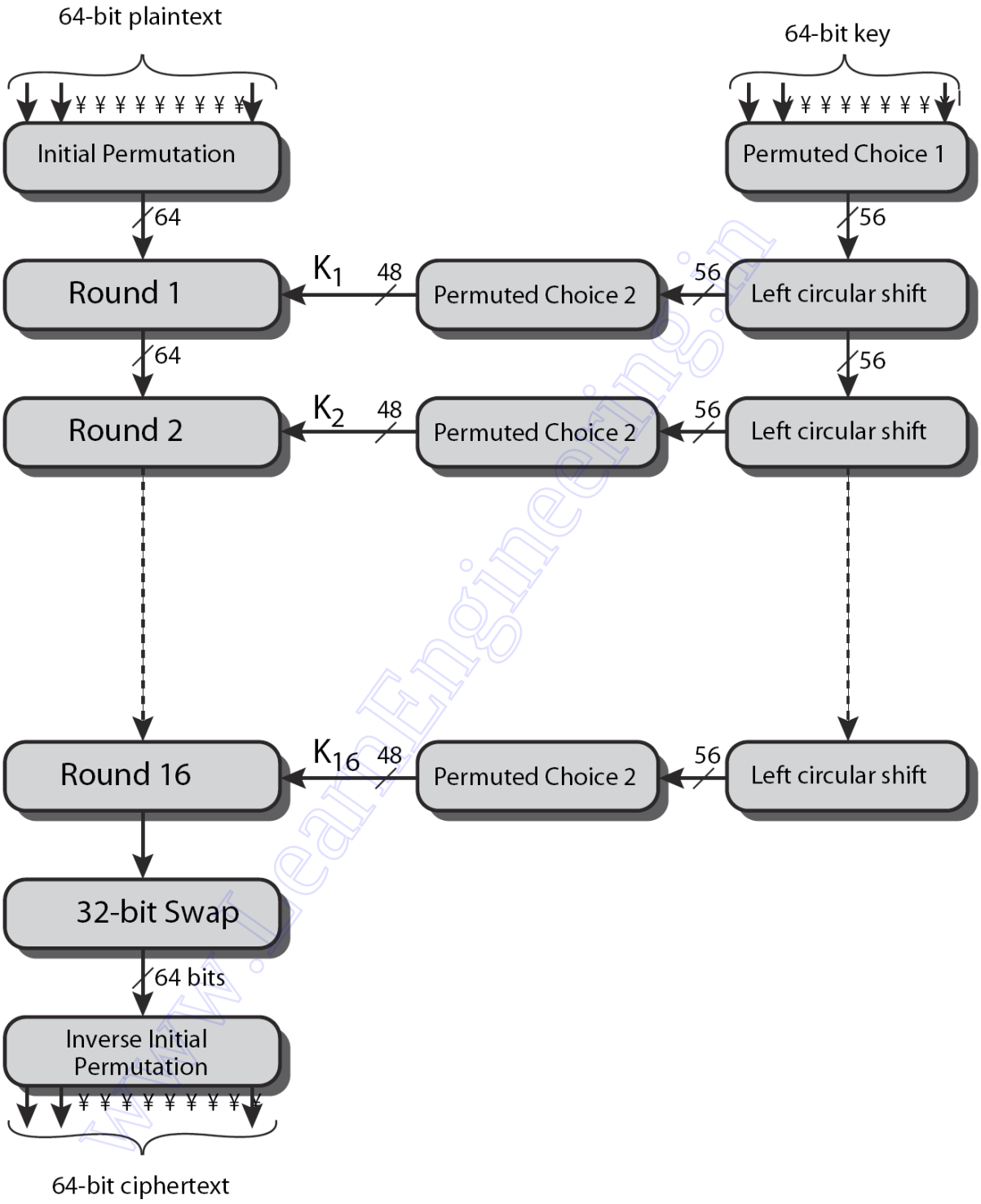


Figure: DES Encryption Algorithm

of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that

are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the preoutput. Finally, the preoutput is passed through a permutation [IP^{-1}] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.

With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

The right-hand portion shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a *subkey*(*Ki*) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

Initial Permutation

The initial permutation and its inverse are defined by tables:

(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

The input to a table consists of 64 bits numbered from 1 to 64.The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

These two permutation functions are indeed the inverse of each other, consider the following 64-bit input :

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

whereM_i is a binary digit. Then the permutation X=IP(M) is as follows:

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1

M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

If we then take the inverse permutation $Y = \text{IP}^{-1}(X) = \text{IP}^{-1}(\text{IP}(M))$, it can be seen that the original ordering of the bits is restored.

Details of Single Round

The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

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

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

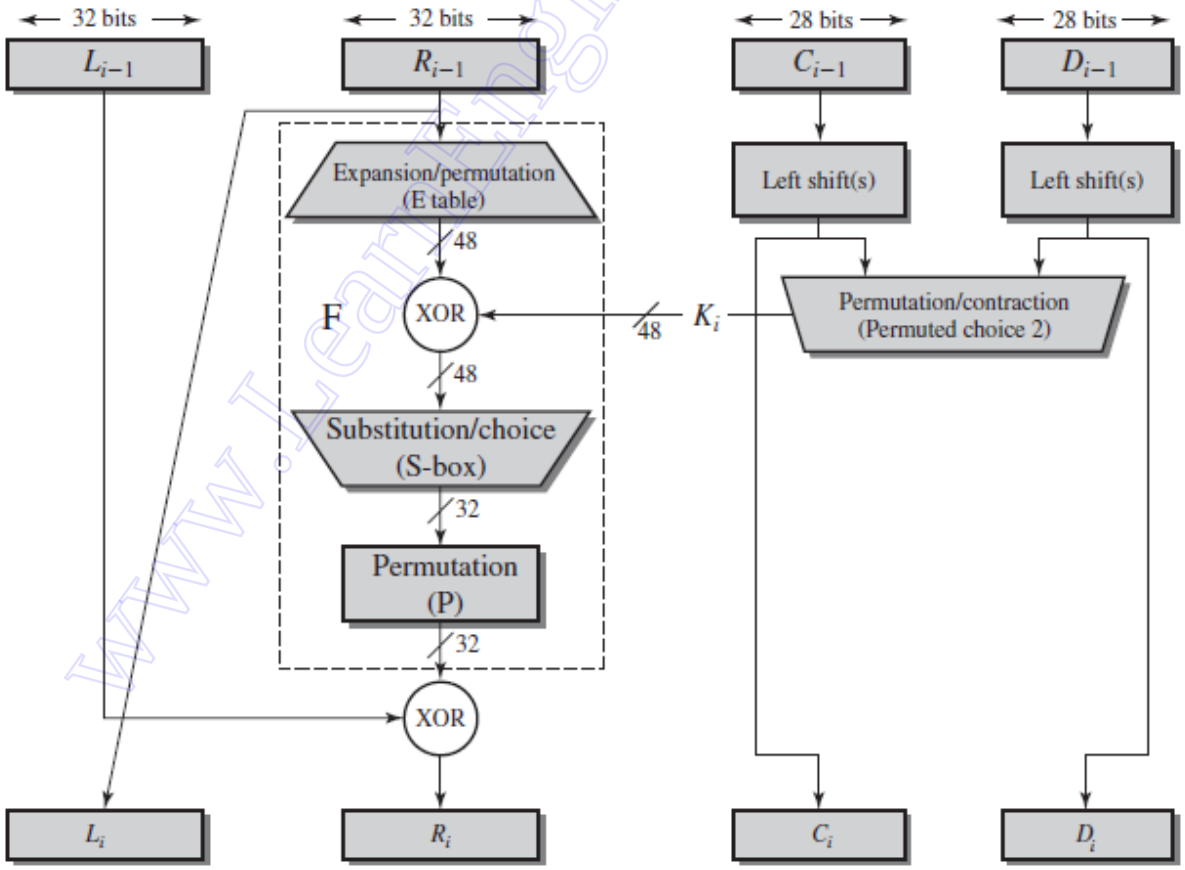


Figure: Single round of DES Algorithm

The round key is 48 bits. The input is 32 bits. This input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the bits.

(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

The resulting 48 bits are XORedwith . This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as:

(d) Permutation Function (P)

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

The role of the S-boxes in the function F is illustrated as:

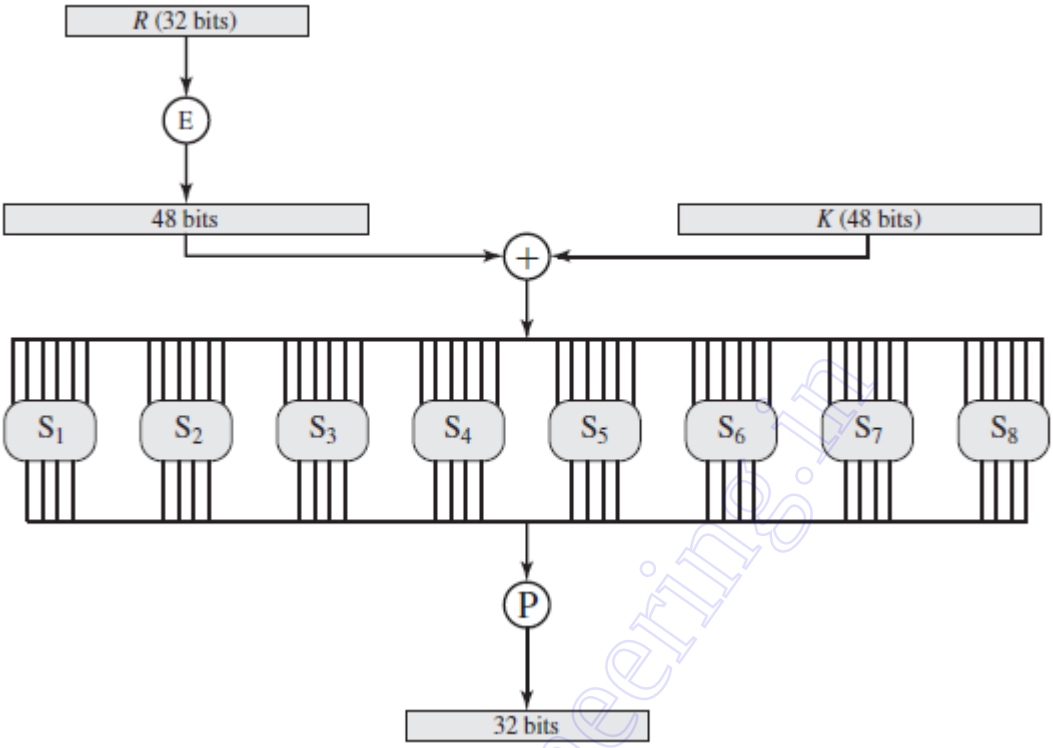


Figure: Calculation of F(R,K)

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

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

Table: Definition of DES S-boxes

The first and last bits of the input to box S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i. The middle four bits select one of the sixteen columns.

The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example, in S_1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12).The value in row 1, column 12 is 9, so the output is 1001.

Each row of an S-box defines a general reversible substitution. For example, if part of the input word is

... efghijklmnop ...

this becomes

... defghihijklmlmnopq ...

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

Key Generation

A 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored.The key is first subjected to a permutation governed by a table labeled Permuted Choice One (PC-1). The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0 . At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift or (rotation) of 1 or 2 bits. These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two (PC-2), which produces a 48-bit output that serves as input to the function $F(R_{i-1},K_i)$.

(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

Table: DES Key Schedule Calculation

DES Decryption

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

The Avalanche Effect

A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext. In particular, a change in one bit of the plaintext or one bit of the key should produce a

change in many bits of the ciphertext. If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched. DES exhibits a strong avalanche effect.

Table: Avalanche Effect in DES

(a) Change in Plaintext		(b) Change in Key	
Round	**Number of bits that differ**	**Round**	**Number of bits that differ**
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

In (a), two plaintexts that differ by one bit were used:

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

with the key

0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

After just three rounds, 21 bits differ between the two blocks. On completion, the two ciphertexts differ in 34 bit positions.

In (b), a single plaintext input:

01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

with two keys that differ in only one bit position:

1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Again, the results show that about half of the bits in the ciphertext differ and that the avalanche effect is pronounced after just a few rounds.

The Strength of Des

The concerns about the level of security provided by DES, fall into two areas:

- key size and
- Nature of the algorithm.

The Use of 56-Bit Keys

With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 x 1016. Thus, on the face of it, a brute-force attack appears impractical. On average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

The Nature of the DES Algorithm

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

Timing Attacks

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

2. RSA ALGORITHM

- **In a public key system using RSA, you intercept the cipher text C=10 sent to a user whose public key is e=5, n=35. What is the plain text? Explain the above problem with an algorithm description. (16) (MAY/JUNE 2012)**
- **Explain RSA algorithm with example as: p=11, q=5, e=3 and PT=9 (16)**

(NOV/DEC 2013)

- **Demonstrate encryption and decryption for the RSA algorithm parameters: p=3, q=11, e=7, d=? , M=5. (8) (MAY/JUNE 2014)**
- **Describe the mathematical foundations of RSA algorithm . Perform the encryption and decryption for the following: p=17, q=7, e=5, n=119, message="6". Use Extended Euclid's algorithm to find the private key. (16)**

(NOV/ DEC 2014)

- **Explain the RSA algorithm in detail. For the given values, trace the sequence of calculations in RSA. p=7, q=13, e=5 and M=10. (16)**

(APR/MAY 2015) (MAY/JUNE 2014) (APR/MAY 2011) (NOV/DEC 2011) (MAY/JUNE 2013) (NOV/DEC 2012)

One of the first successful algorithm to the public key cryptography was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The Rivest-Shamir-Adleman (RSA) scheme is the most widely accepted and implemented general-purpose approach to public-key encryption.

The **RSA** scheme is a cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}.

Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to $\log_2(n) + 1$.

Encryption and decryption for some plaintext block M and ciphertext block C are:

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

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

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$.

For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of e, d, and n such that $M^{ed} \bmod n = M$ for all $M < n$.

2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.

3. It is infeasible to determine d given e and n.

By the first requirement, there is a need to find a relationship of the form

$$M^{ed} \bmod n = M$$

The preceding relationship holds if e and d are multiplicative inverses modulo $\phi(n)$, where $\phi(n)$ is the Euler totient function. For any prime numbers p, q, $\phi(pq) = (p - 1)(q - 1)$. The relationship between e and d can be expressed as

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

This is equivalent to saying

$$ed \equiv 1 \; mod \; \phi(n)$$

$$d \equiv e^{-1} mod \; \phi(n)$$

That is, e and d are multiplicative inverses mod f(n). Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\phi(n)$. Equivalently, $\gcd(\phi(n), d) = 1$.

The ingredients of the RSA schene are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with gcd($\phi(n)$, e) = 1; 1 <e <$\phi(n)$	(public, chosen)
$d \equiv e^{-1} mod\ \phi(n)$	(private, calculated)

The private key consists of {d, n} and the public key consists of {e, n}. Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates C = M^e mod n and transmits C. On receipt of this ciphertext, user A decrypts by calculating M = C^d mod n.

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

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure: The RSA Algorithm

Example:

The keys were generated as follows.

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

2.Calculate $n = pq$= 17 * 11 = 187.

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

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

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

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}.

Assume M = 88,

For encryption,

Calculate C = 887 mod 187. Exploiting the properties of modular arithmetic, this can be done as follows.

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

88^1 mod 187 = 88

88^2 mod 187 = 7744 mod 187 = 77

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

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

For decryption,

Calculate $M = 11^{23}$ mod 187:

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

11^1 mod 187 = 11

11^2 mod 187 = 121

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

11^8 mod 187 = 214,358,881 mod 187 = 33

11^{23} mod 187 = (11 × 121 × 55 × 33 × 33) mod 187

= 79,720,245 mod 187 = 88

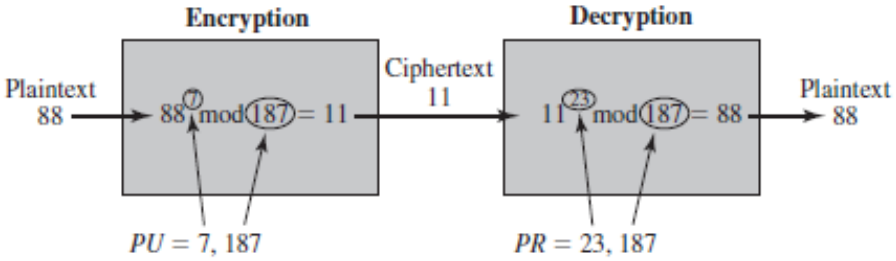


Figure: Example of RSA Algorithm

Computational Aspects

There are actually two issues to consider:

- encryption/decryption and
- key generation.

Exponentiation in Modular Arithmetic

Both encryption and decryption in RSA involve raising an integer to an integer power, mod n. If the exponentiation is done over the integers and then reduced modulo n, the intermediate values would be gargantuan. The property of modular arithmetic is:

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

Intermediate results modulo n can be reduced.

Another consideration is the efficiency of exponentiation, because with RSA, we are dealing with potentially large exponents. Consider compute x^{16}. A straightforward approach requires 15 multiplications:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

More generally, suppose we wish to find the value $a^b \bmod n$ with a, b, and m positive integers. If we express b as a binary number $b_k b_{k-1} \ldots b_0$, then we have

$$B = \sum_{b_t \neq 0} 2^i$$

Therefore,

$$a^b = a^{(\Sigma_{b_t \neq 0}\, 2^t)} = \prod_{b_t \neq 0} a^{(2^t)}$$

$$a^b \bmod n = \left[\prod_{b_t \neq 0} a^{(2^t)}\right] \bmod n = \left(\prod_{b_t \neq 0} [a^{(2^t)} \bmod n]\right) \bmod n$$

```
                    c ← 0; f ← 1
for i←k down to 0
        do c ← 2 × c
           f ← (f × f) mod n
        if b_i = 1
          then c ← c + 1
                  f ← ( f × a) mod n
return f
```

Figure: Algorithm for computing a^b mod n

Key Generation

Before the application of the public-key cryptosystem, each participant must generate a pair of keys. This involves the following tasks.

• Determining two prime numbers, *p* and *q*.

• Selecting either *e* or *d* and calculating the other.

Primes p and q must be chosen from a sufficiently large set. The procedure that is generally used is to pick at random an odd number of the desired order of magnitude and test whether that number is prime. If not, pick successive random numbers until one is found that tests prime.

The procedure for picking a prime number is as follows.

1. Pick an odd integer *n* at random (e.g., using a pseudorandom number generator).
2. Pick an integer *a* <*n* at random.
3. Perform the probabilistic primality test, such as Miller-Rabin, with *a* as a parameter. If *n* fails the test, reject the value *n* and go to step 1.
4. If *n* has passed a sufficient number of tests, accept *n*; otherwise, go to step 2.

The Security of RSA (MAY/JUNE 2007)

Five possible approaches to attacking the RSA algorithm are

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

The Factoring Problem (MAY/JUNE 2012)

Three approaches can be identified to attacking RSA mathematically.

1. Factor n into its two prime factors. This enables calculation of $\phi(n)$ = (p - 1) × (q - 1), which in turn enables determination of $d \equiv e^{-1}(mod\ \phi(n))$.
2. Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d \equiv e^{-1}(mod\ \phi(n))$.
3. Determine d directly, without first determining $\phi(n)$.

3. DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

- **Explain Diffie-Hellman Key exchange algorithm with its merits and demerits. (APR/MAY 2011) (MAY/JUNE 2014) (APR/MAY 2010) (MAY/JUNE 2013) (NOV/DEC 2012)**
- **Users A and B use the Diffie-Hellman Key exachange technique with a common prime q=71 and a primitive root $\alpha = 7$. If the user A has private key X_A=5, what is A's public key Y_A? (8) (MAY/JUNE 2014)**
- **Explain Diffie-Hellman key exchange algorithm with an example. Consider a Diffie-Hellman scheme with a common prime q=353 and a primitive root α=3. Users A and B have private keys X_A=17 and X_B=21 respectively. What is the shared secret key K_1 and K_2? (16) (NOV-DEC 2014)**

❖ **How does Diffie-Hellman key exchange achieve security? (2) (MAY/JUNE 2007)**

The first published public-key algorithm that define public-key cryptography is generally referred to as Diffie-Hellman key exchange.

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

A primitive root of a prime number whose powers modulo p generate all the integers from 1 to p -1 .That is, if a is a primitive root of the prime number p , then the numbers

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

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 , a unique exponent can be found such that

$$b = a^i (mod\ p) where \quad 0 \leq i \leq (p-1)$$

The exponent is referred to as the **discrete logarithm** of b for the base a , mod p .This value can be expressed as $dlog_{a,p}(b)$.

The Algorithm

For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q.

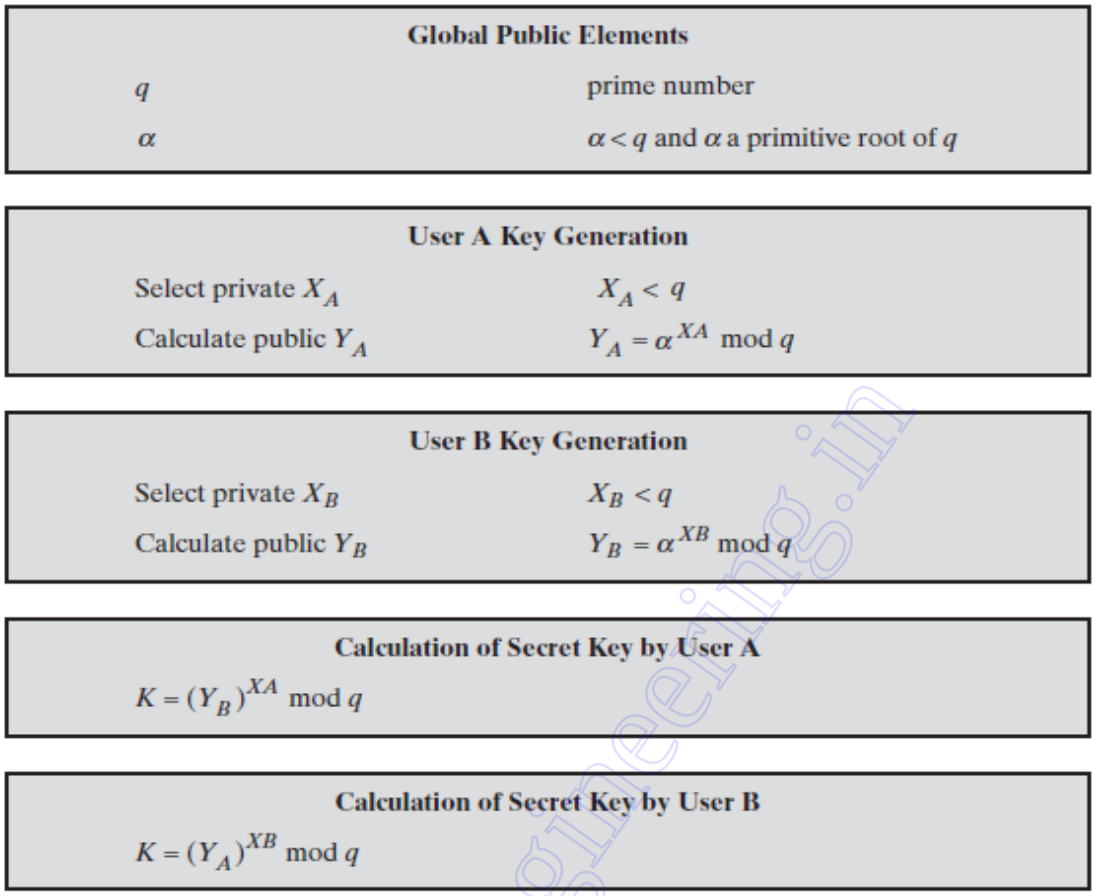


Figure: The Diffie-Hellman Key Exchange Algorithm

. Suppose the users A(Alice) and B(Bob) wish to exchange a key. User A selects a random integer X_A<q and computes $Y_A = \alpha^{X_A} mod\ q$. Similarly, user B independently selects a random integer X_B<q and computes $Y_B = \alpha^{X_B} mod\ q$. Each side keeps the X value private and makes the Y value available publicly to the other side. Thus, $X_A$is A's private key and $Y_A$is A's corresponding public key, and similarly for B. User A computes the key as $K = (Y_B)^{X_A} mod\ q$ and user B computes the key as $K = (Y_A)^{X_B} mod\ q$. These two calculations produce identical results:

(by the rules of modular arithmetic)

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

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

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

$$= \alpha^{X_B X_A} mod\ q$$

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

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

$$= (Y_A)^{X_B} mod\ q$$

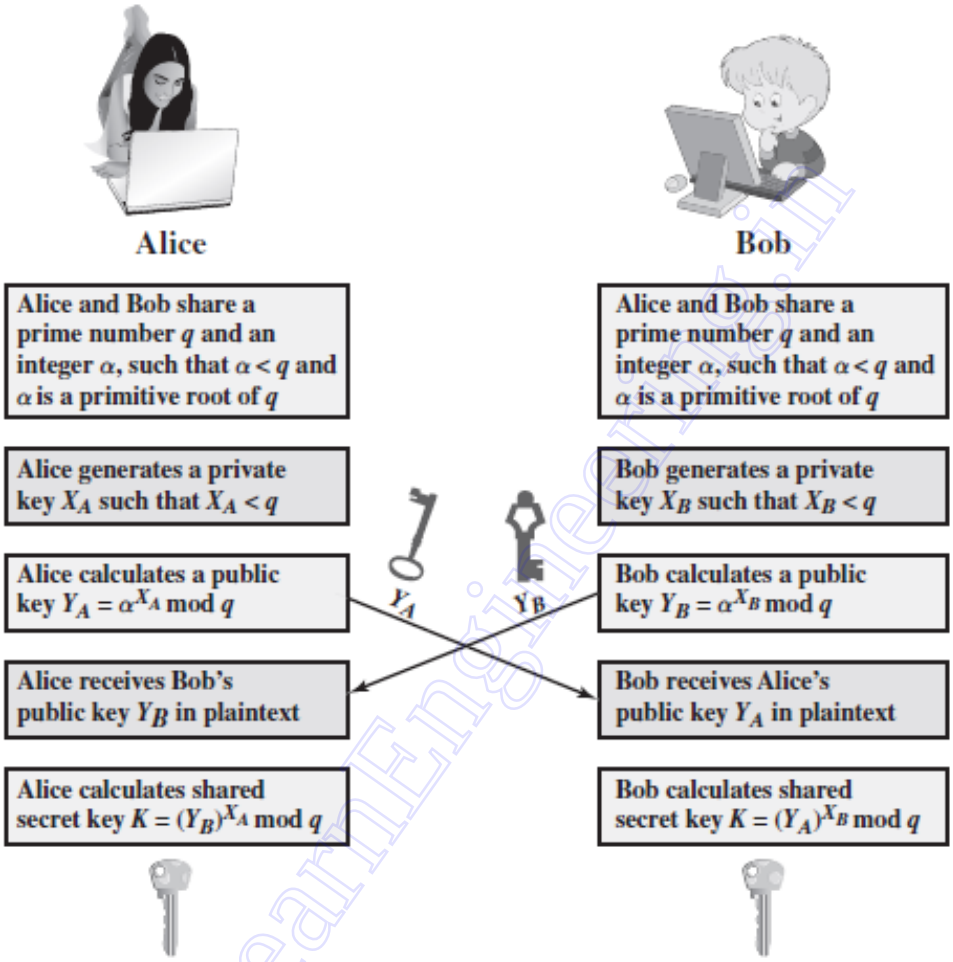


Figure: The Diffie-Hellman Key Exchange

The result is that the two sides have exchanged a secret value. Typically, this secret value is used as shared symmetric secret key.

Because X_A and X_B are private, an adversary only has the following ingredients to work with: q, α, Y_A, and Y_B. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

$$X_B = d\ log_{\alpha,q}(Y_B)$$

The adversary can then calculate the key *K* in the same manner as user B calculates it. That is, the adversary can calculate *K* as

$$K = (Y_A)^{X_B} mod\ q$$

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Example:

Key exchange is based on the use of the prime number q = 353 and a primitive root of 353, in this case α = 3. A and B select private keys X_A= 97 and X_B= 233, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.

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

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

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

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

In this simple example, it would be possible by brute force to determine the secret key 160.

Key Exchange Protocols

A simple protocol that makes use of the Diffie-Hellman calculation is:

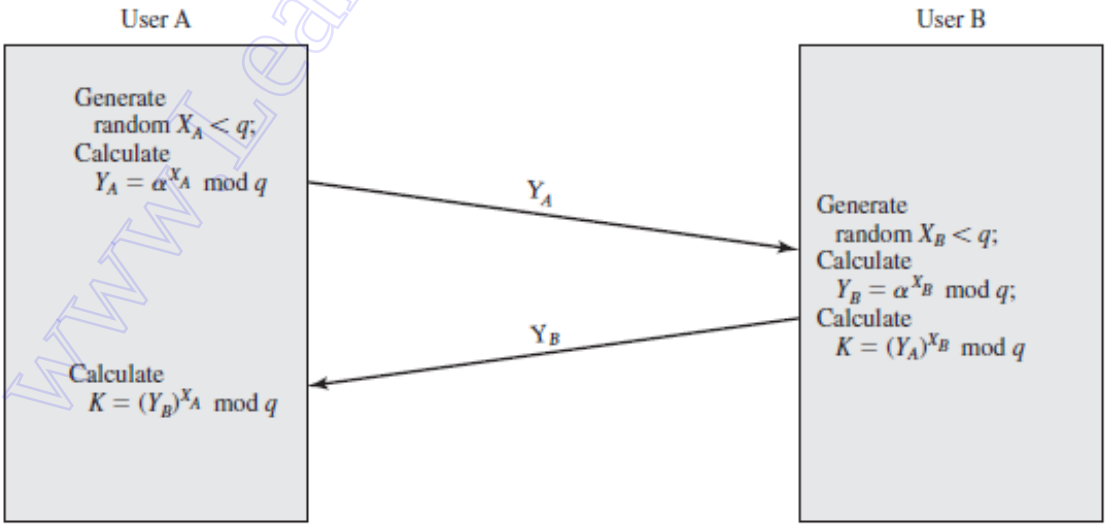


Figure: Diffie-Hellman Key Exchange

Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A, calculate Y_A , and send that to user B. User B responds by generating a private value X_B , calculating Y_B, and sending to user A. Both users can now calculate the key. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message. This technique does not protect against replay attacks.

Man-in-the-Middle Attack

The protocol is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

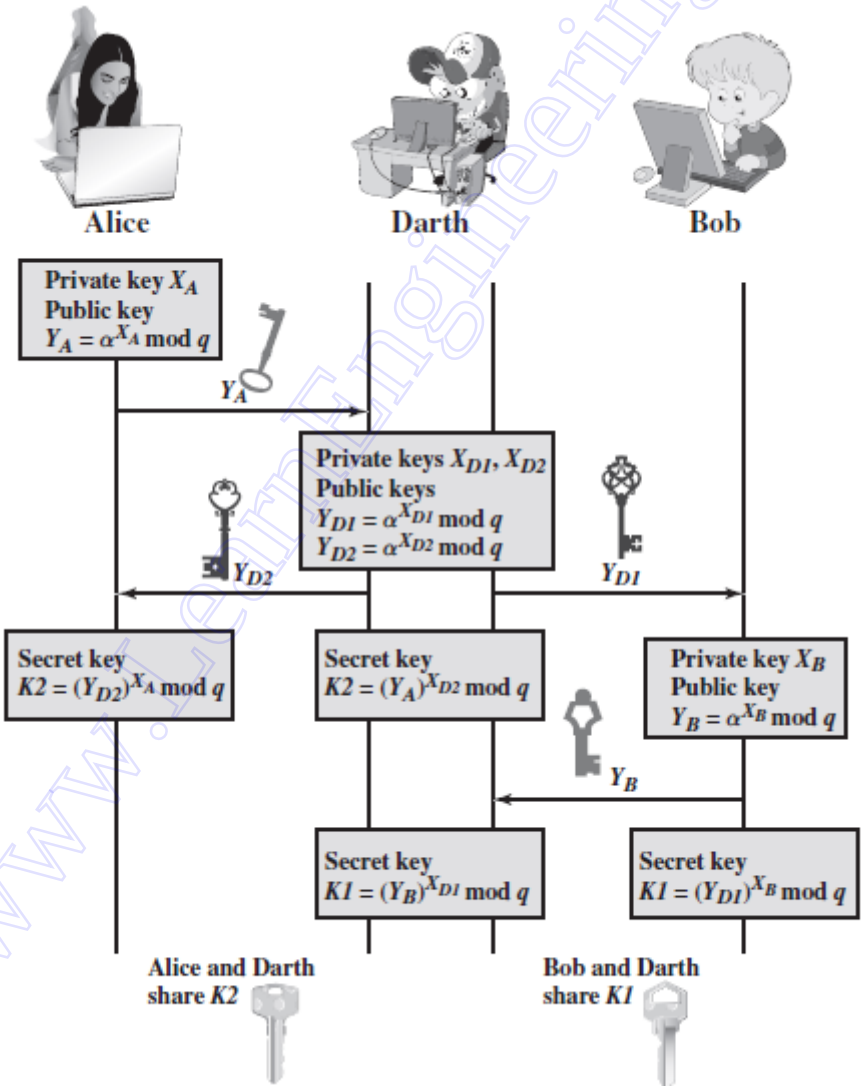


Figure: Man-in-the-middle attack

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates
$$K2 = (Y_A)^{X_{D2}} mod\ q$$
4. Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{X_B} mod\ q$.
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates $K1 = (Y_B)^{X_{D1}} mod\ q$.
7. Alice receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A} mod\ q$.

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.

4. ELLIPTIC CURVE CRYPTOGRAPHY

❖ **Explain how the elliptic curves are useful for cryptography? (16)**
(MAY/JUNE 2012)

❖ **Using Elliptic curve encryption/decryption scheme, key exchange between users A and B is accomplished. The cryptosystem parameters are, Ellyptic group of points E11 (1,6) and point G on the elliptic curve is G=(2,7). B's secret key is nB = 7. Now when. (i) A wishes to encrypt the message Pm = (10,9) and chooses the random value K=3. Determine the ciphertext Cm. (ii) How will B recover Pm form Cm. (iii) Find out B's public key PB. (16) (MAY/JUNE 2007)**

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

User A Key Generation	
Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation	
Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A
$K = n_A \times P_B$

Calculation of Secret Key by User B
$K = n_B \times P_A$

Figure: ECC Diffie-Hellman Key Exchange

Key exchange using elliptic curves can be done as:

- First pick a large integer q, which is either a prime number p or an integer of the form 2^m, and elliptic curve parameters a and b. This defines the elliptic group of points $E_q(a, b)$.
- Next, pick a *base point* $G = (x_1, y_1)$ in $E_p(a, b)$ whose order is a very large value n. The order n of a point G on an elliptic curve is the smallest positive integer n such that $nG = 0$ and G are parameters of the cryptosystem known to all participants.
- A key exchange between users A and B can be accomplished as:
    - A selects an integer n_A less than n. This is A's private key. A then generates a public key $P_A = n_A \times G$; the public key is a point in $E_q(a, b)$.
    - B similarly selects a private key n_B and computes a public key P_B.
    - A generates the secret key $k = n_A \times P_B$. B generates the secret key $k = n_B \times P_A$.

- The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

- To break this scheme, an attacker would need to be able to compute k given G and kG, which is assumed to be hard.

Example:

Consider p = 211; $E_p(0,-4)$, which is equivalent to the curve $y_2 = x_3 - 4$; and G = (2, 2). One can calculate that $240G = O$. A's private key is n_A= 121, so A's public key is P_A= 121(2, 2) = (115, 48). B's private key is n_B= 203, so B's public key is 203(2, 3) = (130, 203). The shared secret key is 121(130, 203) = 203(115, 48) = (161, 69).

- The secret key is a pair of numbers. If this key is to be used as a session key for conventional encryption, then a single number must be generated.

Elliptic Curve Encryption/Decryption

The first task in this system is to encode the plaintext message m to be sent as an (x, y) point P_m.

As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key $nA$and generates a public key $P_A = n_A \times G$.

To encrypt and send a message $P_m$to B, A chooses a random positive integer k and produces the ciphertext$C_m$consisting of the pair of points:

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

To decrypt the ciphertext, B multiplies the first point in the pair by B's private key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

A has masked the message Pm by adding $kP_B$to it. Nobody but A knows the value of k, so even though $P_b$is a public key, nobody can remove the mask kP_B.

For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

Example:

The global public elements are q = 257; Eq(a, b) = E_{257}(0, -4), which is equivalent to the curve $y_2 = x_3$ - 4; and G = (2, 2). Bob's private key is n_B= 101, and his public key is P_B= $n_B G$= 101(2, 2) = (197, 167). Alice wishes to send a message to Bob that is encoded in the elliptic point P_m= (112, 26). Alice chooses random integer k = 41 and computes kG= 41(2, 2) = (136, 128), kP_B= 41(197, 167) = (68, 84) and P_m+ kP_B= (112, 26) + (68, 84) = (246, 174). Alice sends the ciphertextC_m= (C_1, C_2) = {(136, 128), (246, 174)} to Bob. Bob receives the ciphertext and computes C_2 - $n_B C_1$ = (246, 174) - 101(136, 128) = (246, 174) - (68, 84) = (112, 26).

Security of Elliptic Curve Cryptography

The security of ECC depends on how difficult it is to determine k given $kP$and P. This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the Pollard rho method. A considerably smaller key size can be used for ECC compared to RSA. Furthermore, for equal key lengths, the computational effort required for ECC and RSA is comparable. Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA.

5. BLOCK CIPHER MODES OF OPERATION

❖ **Explain the Block cipher modes of operation. (16) (APR/MAY 2010, NOV/DEC 2013, APR/MAY 2011)**

❖ **What is the disadvantage with ECB mode of operation? (2) (MAY/JUNE 2013)**

A block cipher algorithm is a basic building block for providing data security. To apply a block cipher in a variety of applications, four "modes of operation" have been defined by NIST. In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream.

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

1.Electronic Codebook Mode

The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key.

The term *codebook* is used because, for a given key, there is a unique ciphertext for every *b*-bit block of plaintext. For a message longer than *b* bits, the

procedure is simply to break the message into *b*-bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key.

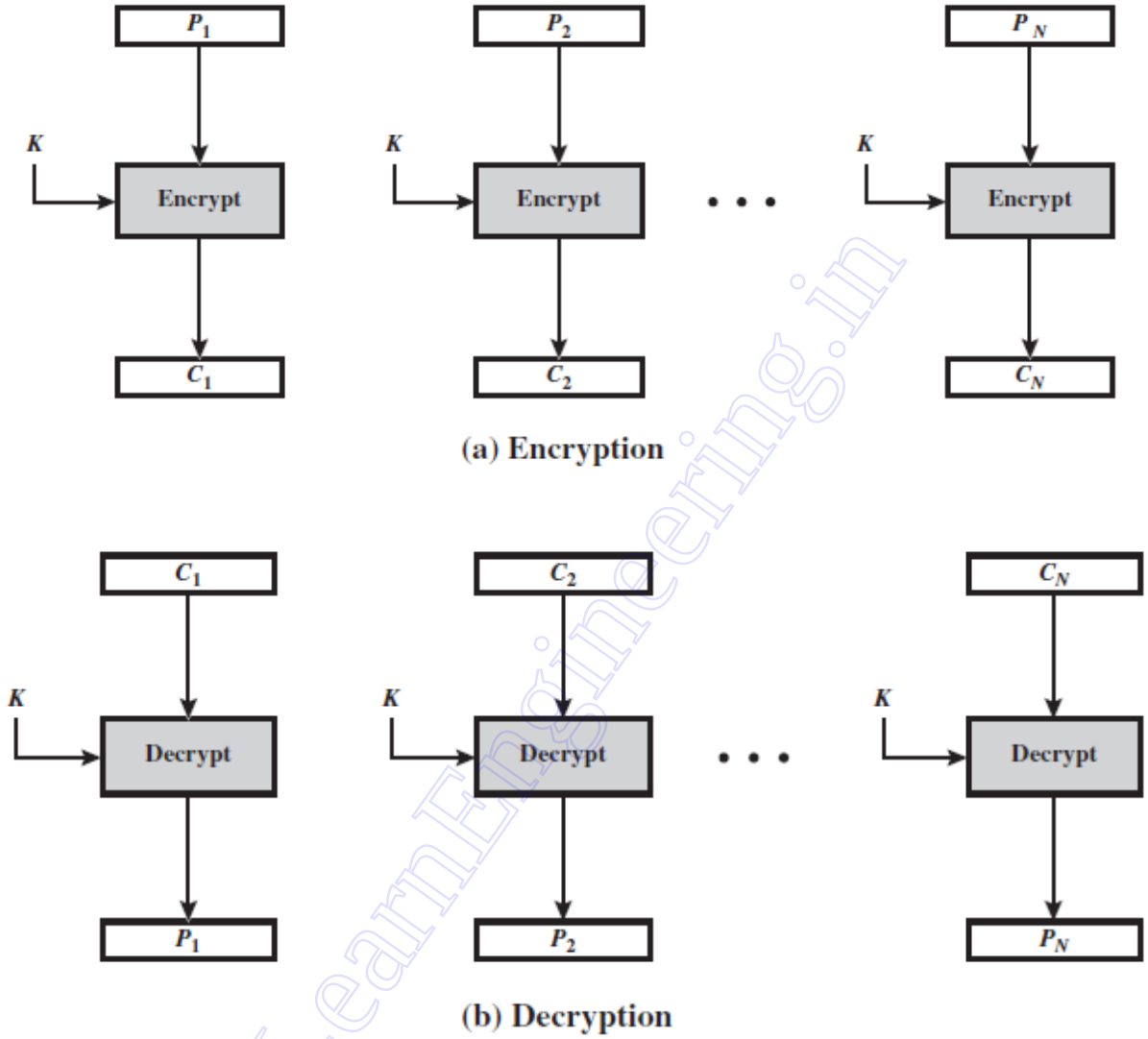


Figure: Electronic Code Book (ECB) mode

The plaintext (padded as necessary) consists of a sequence of *b*-bit blocks, P_1, P_2,...,P_N; the corresponding sequence of ciphertext blocks is C_1, C_2,..., C_N. The ECB mode is defined as:

ECB	$C_j = E(K, P_j) \qquad j = 1, \ldots, N$	$P_j = D(K, C_j) \qquad j = 1, \ldots, N$

The most significant characteristic of ECB is that the same *b*-bit block of plaintext, if it appears more than once in the message, always produces the same ciphertext.

Disadvantage

The ECB method is ideal for a short amount of data, such as an encryption key. For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.

2.Cipher Block Chaining Mode

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode.

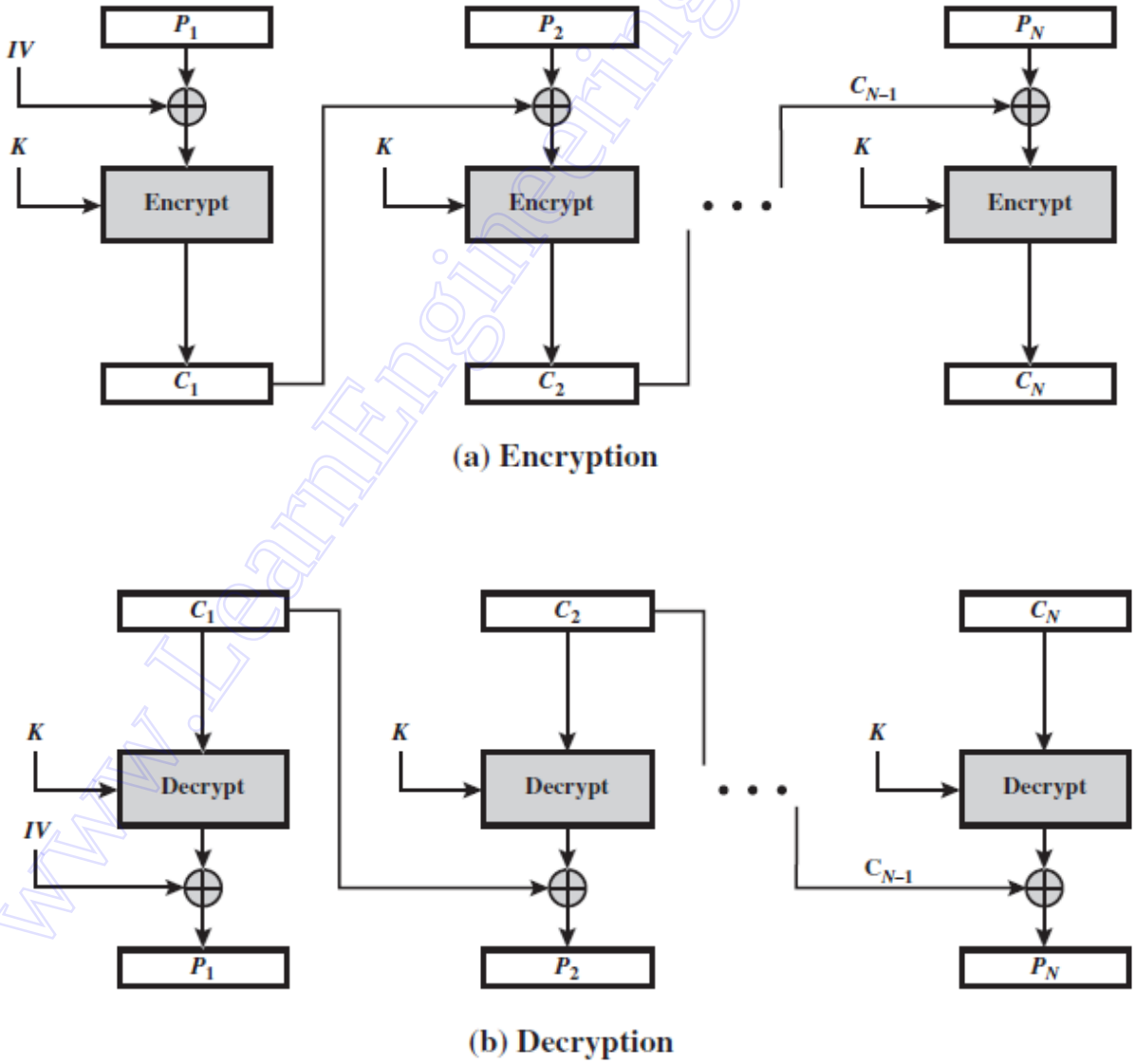


Figure: Cipher Block Chaining (CBC) mode

In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of *b* bits are not exposed. As with the ECB mode, the CBC mode requires that the last block be padded to a full *b* bits if it is a partial block.

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

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

Then

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

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

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

To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext. The IV is a data block that is the same size as the cipher block. The CBC mode can be defined as

CBC	$C_1 = E(K, [P_1 + IV])$ $C_j = E\left(K, \left[P_j \oplus C_{j-1}\right]\right) \quad j = 2, \ldots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D\left(K, C_j\right) \oplus C_{j-1} \quad j = 2, \ldots., N$

The IV must be known to both the sender and receiver but be unpredictable by a third party. For maximum security, the IV should be protected against unauthorized changes. This could be done by sending the IV using ECB encryption. One reason for protecting the IV is as follows: If an opponent is able to fool the receiver into using a different value for IV, then the opponent is able to invert selected bits in the first block of plaintext.

Two possible methods of protecting IV:

- The first method is to apply the encryption function, under the same key that is used for the encryption of the plaintext, to a **nonce**. The nonce must be a data block that is unique to each execution of the encryption operation.

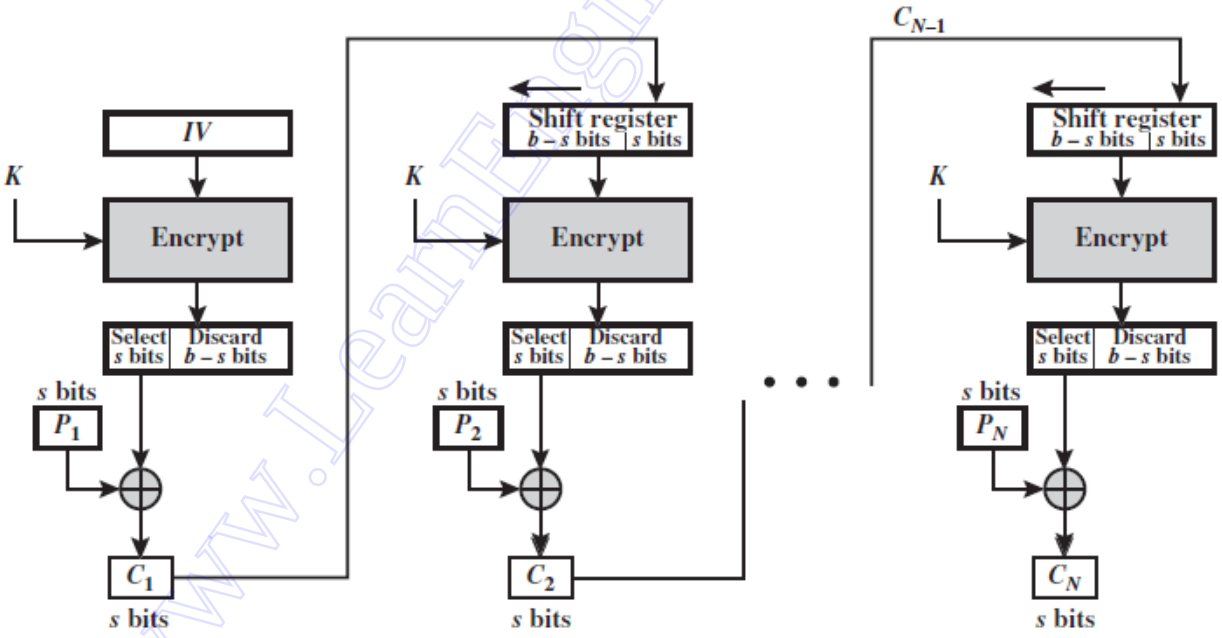
- The second method is to generate a random data block using a random number generator.

Because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than *b* bits. In addition to its use to achieve confidentiality, the CBC mode can be used for authentication.

3.Cipher Feedback Mode

The DES scheme is essentially a block cipher technique that uses *b*-bit blocks. However, it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a cipher text output of 8 bits.



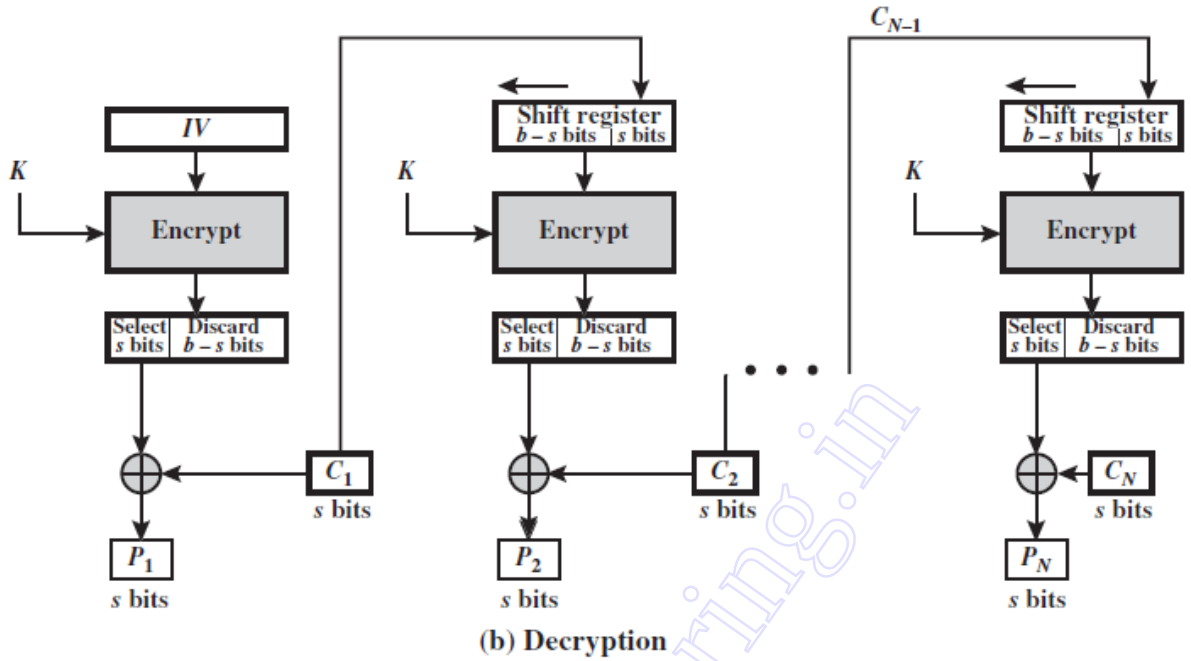


Figure: s-bit Cipher Feedback (CFB) mode

As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than units of *b* bits, the plaintext is divided into *segments* of *s* bits.

For encryption, The input to the encryption function is a *b*-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) *s* bits of the output of the encryption function are XORed with the first segment of plaintext *P*1 to produce the first unit of ciphertext*C*1, which is then transmitted. In addition, the contents of the shift register are shifted left by *s* bits, and *C*1 is placed in the rightmost (least significant) *s* bits of the shift register. This process continues until all plaintext units have been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Let $MSB_s(X)$ be defined as the most significant *s* bits of *X*. Then

$$C_1 = P_1 \oplus MSB_s[E(K, IV)]$$

Therefore, by rearranging terms:

$$P_1 = C_1 \oplus MSB_s[E(K, IV)]$$

The same reasoning holds for subsequent steps in the process.

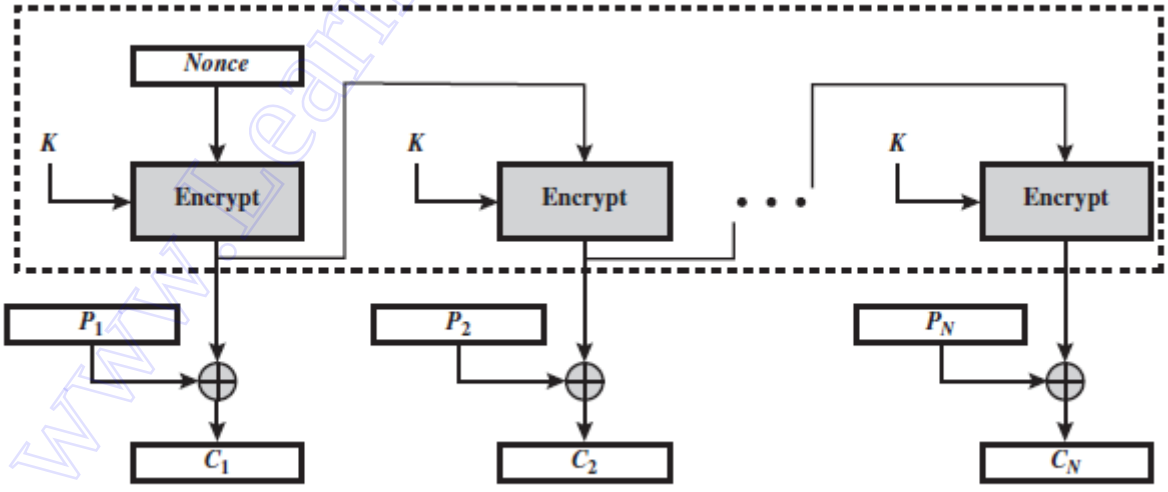
CFB mode can be defined as:

CFB	$I_1 = IV$ $I_j = LSB_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = E(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus MSB_s(O_j) \quad j = 1, \ldots, N$	$I_1 = IV$ $I_j = LSB_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = E(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus MSB_s(O_j) \quad j = 1, \ldots, N$

In the case of CFB, the stream of bits that is XORed with the plaintext also depends on the plaintext. In CFB encryption, like CBC encryption, the input block to each forward cipher function (except the first) depends on the result of the previous forward cipher function; therefore, multiple forward cipher operations cannot be performed in parallel. In CFB decryption, the required forward cipher operations can be performed in parallel if the input blocks are first constructed (in series) from the IV and the ciphertext.

4. Output Feedback Mode

The **output feedback** (OFB) mode is similar in structure to that of CFB. For OFB, the output of the encryption function is fed back to become the input for encrypting the next block of plaintext.



(a) Encryption

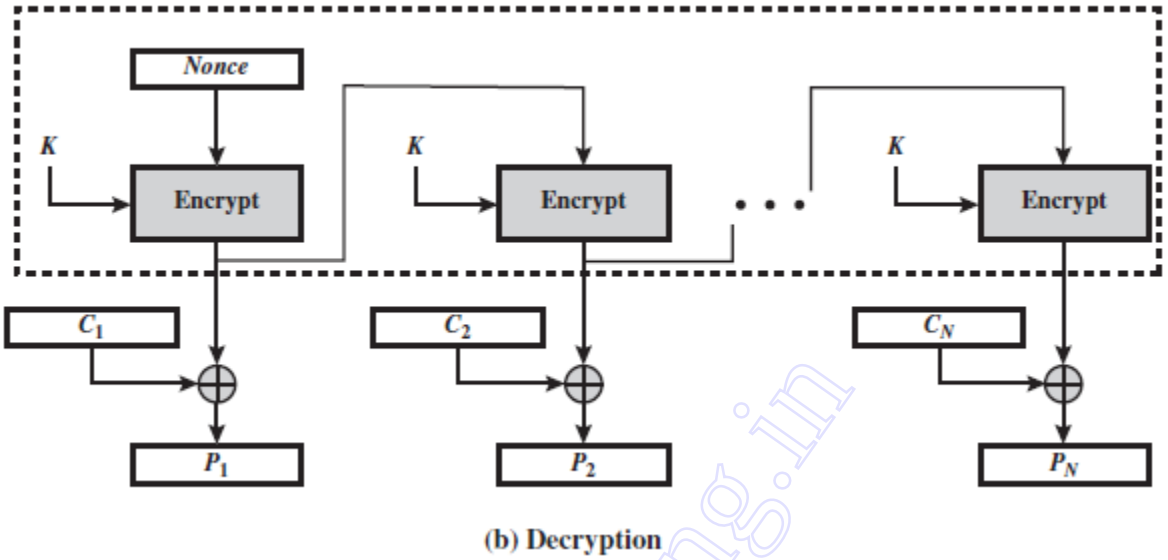


Figure: Output Feedback (OFB) mode

In CFB, the output of the XOR unit is fed back to become input for encrypting the next block. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, whereas CFB operates on an *s*-bit subset. OFB encryption can be expressed as

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

Where,

$$O_{j-1} = E(K, O_{j-2})$$

The encryption can be defined as:

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

and decryption is

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

OFB mode can be defined as:

OFB	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = E(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C_N = P_N \oplus MSB_u(O_N)$	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = E(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P_N = C_N \oplus MSB_u(O_N)$

As with CBC and CFB, the OFB mode requires an initialization vector. In the case of OFB, the IV must be a nonce; that is, the IV must be unique to each execution of the encryption operation. The reason for this is that the sequence of encryption output blocks, *Oi*, depends only on the key and the IV and does not depend on the plaintext. Therefore, for a given key and IV, the stream of output bits used to XOR with the stream of plaintext bits is fixed.

One advantage of the OFB method is that bit errors in transmission do not propagate. For example, if a bit error occurs in C_1, only the recovered value of P_1 is affected; subsequent plaintext units are not corrupted. With CFB, C_1 also serves as input to the shift register and therefore causes additional corruption downstream.

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

5. Counter Mode

Counter (CTR) mode has applications to ATM (asynchronous transfer mode) network security and IPSec (IP security).

A counter equal to the plaintext block size is used. the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^b, where b is the block size).

For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block. Thus, the initial counter value must be made available for decryption.

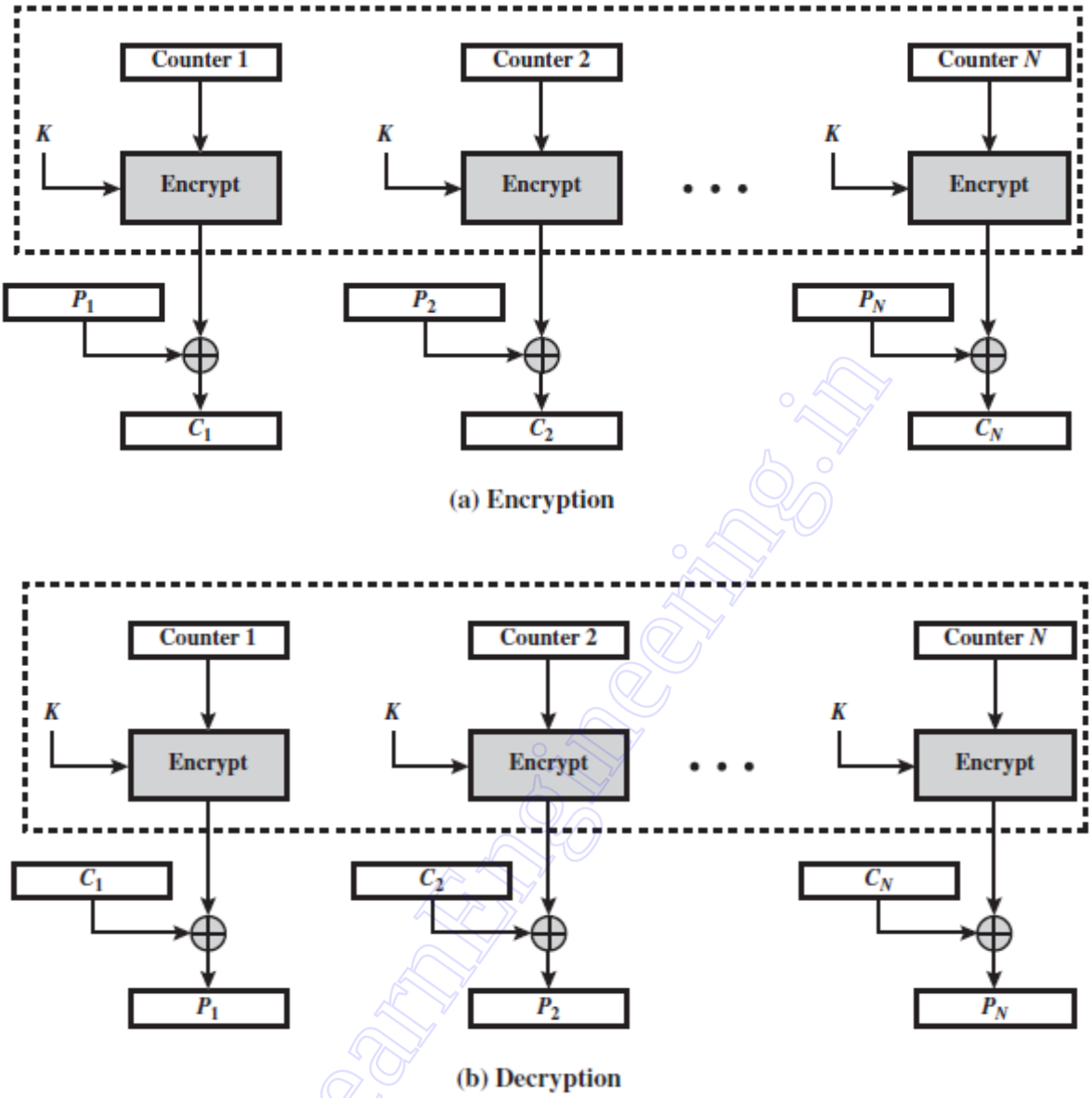


Figure: Counter (CTR) mode

Given a sequence of counters T_1, T_2, …,T_N, CTR mode can be defined as:

CTR	$C_j = P_j \oplus E(K, T_j) \quad j = 1, \ldots, N-1$ $C_N = P_N \oplus MSB_u[E(K, T_N)]$	$P_j = C_j \oplus E(K, T_j) \quad j = 1, \ldots, N-1$ $P_N = C_N \oplus MSB_u[E(K, T_N)]$

For the last plaintext block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the XOR operation; the remaining b - u bits are discarded.

As with the OFB mode, the initial counter value must be a nonce; that is, T_1 must be different for all of the messages encrypted using the same key. Further, all T_i values across all messages must be unique.

One way to ensure the uniqueness of counter values is to continue to increment the counter value by 1 across messages. That is, the first counter value of the each message is one more than the last counter value of the preceding message.

Advantages

- **Hardware efficiency:** Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved.
- **Software efficiency:** Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions, can be effectively utilized.
- **Preprocessing:** If sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions. When the plaintext or ciphertext input is presented, then the only computation is a series of XORs. Such a strategy greatly enhances throughput.
- **Random access:** The *i*th block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block *Ci*cannot be computed until the *i*- 1 prior block are computed. There may be applications in which a ciphertext is stored and it is desired to decrypt just one block; for such applications, the random access feature is attractive.
- **Provable security:** It can be shown that CTR is at least as secure as the other modes.
- **Simplicity:** Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. In addition, the decryption key scheduling need not be implemented.

UNIT III HASH FUNCTIONS AND DIGITAL SIGNATURES

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC –MD5 - SHA - HMAC – CMAC - Digital signature and authentication protocols – DSS – EI Gamal – Schnorr.

PART – A

1. List the authentication requirements. (May/June'14)
Specify the requirements for message authentication. (May/June'12)

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

2. Define hashing function. (Nov/Dec'13)

Hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M). A hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value.

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

$$h = H(M)$$

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

3. What are the properties of hashing function in cryptography? (Nov/Dec'13 & May/June'11)

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to as the **one-way property**.
5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.

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

4. What do you mean by one-way property in hash function? (April/May'11)

- H is one-way, for any given value h, it is computationally infeasible to find x such that H(x) = h.
- One-way property states that it is easy to generate a code given a message but virtually impossible to generate a message given a code.
- This property is essential for authentication.

5. Write down the purpose of hash function along with a simple hash function. (May/June'07)

- The input (message, file, etc.) is viewed as a sequence of *n*-bit blocks. The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function.
- One of the simplest hash functions is the **bit-by-bit exclusive-OR (XOR)** of every block. This can be expressed as $C_i = b_{i1} \oplus b_{i2} \oplus g \oplus b_{im}$
  where
  C_i = *i*th bit of the hash code, 1 … *i* … n
  m = number of *n*-bit blocks in the input
  b_{ij} = *i*th bit in *j*th block
  $\oplus$ = XOR operation
- This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check.

6. Define message digest. (April/May'10)

When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest**. The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

7. Write any two differences between MD4 and SHA. (Nov/Dec'12)

MD4	SHA
Pad message so its length is 448 mod 512.	Pad message so its length is a multiple of 512 bits.
Initialize the 4-word (128 bit) buffer (A, B, C, D)	Initialize 5-bit (160 bit) buffer (A, B, C, D, E)
Process the message in 16-word chunks using 3 rounds of 16-bit operations each on chunk and buffer.	Process the message in 16-word chunks using 4 rounds of 20-bit operations.

8. What are the performance differences between MD5, SHA-512 and RIPEMD-160? (Nov/Dec'14)

1. MD5 produces a 128-bit hash value. SHA-512 produces 160-bit hash value.
2. Brute force attack harder. (160 like SHA-1 vs 128 bits for MD5)
3. Not vulnerable to known attacks, like SHA-1 though stronger (compared to MD4/5)
4. SHA-512 is slower than MD5 (more steps).
5. All designed as simple and compact.
6. SHA-1 optimized for big-endian CPUs vs RIPMED-160 and MD5 optimized for little-endian CPUs.

9. What are the two important key issues related to authenticated key exchange? (May/June'12)

- Confidentiality
- Timeliness

10. What is digital signature? (April/May'15 & May/June'11)

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. 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 two approaches of digital signature? (Nov/Dec'12)

- Direct Digital Signature
- Arbitrated Digital Signature

12. List any two properties a digital signature should essentially have? (May/June'07)

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

13. What are the requirements of digital signature?

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use information unique to sender to prevent both forgery and denial.
3. It must be relatively easy to produce the digital signature.
4. It must be relatively easy to recognize and verify the digital signature.
5. It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be practical to retain a copy of the digital signature in storage.

14. What are the security services provided by digital signature? (Nov/Dec'14)

- **Authentication -** The assurance that the communicating entity is the one that it claims to be.
- **Data Integrity -** The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
- **Nonrepudiation -** Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

15. What are birthday attacks? (May/June'14)

A birthday attack is a name used to refer to class of brute force attacks. It gets its name from the surprising result that the probability that two or more people in a group of twenty three share the same birthday day is greater than ½ such a result is called birthday paradox.

PART – B

1. MESSAGE AUTHENTICATION FUNCTIONS

- **Discuss in detail about authentication function.**
- **Write about the basic uses of MAC and list out its applications. (May/June'12)**
- **Describe about Hash functions. (Nov/Dec'12)**

a. Message Authentication Functions:

- Message authentication or digital signature mechanism has two levels of functionality.
  - At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
  - This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.
- Three classes of functions that may be used to produce an authenticator.
  - ✓ **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator
  - ✓ **Message encryption:** The ciphertext of the entire message serves as its authenticator
  - ✓ **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

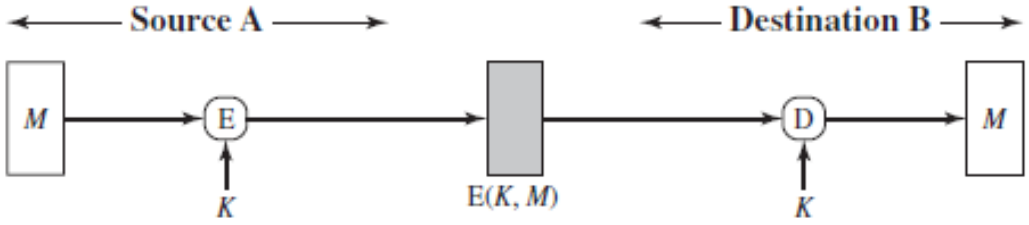
Message Encryption

- Conventional encryption can serve as authenticator
  - Conventional encryption provides *authentication* as well as *confidentiality*
  - Requires recognizable plaintext or other *structure* to distinguish between well-formed legitimate plaintext and meaningless random bits
    - e.g., ASCII text, an appended checksum, or use of layered protocols

Symmetric Encryption / Conventional Encryption

- A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B.
- If no other party knows the key then confidentiality is provided:- No other party can recover the plaintext of the message.

- Given a decryption function D and a secret key K, the destination will accept any input Y and produce output X =D_k(Y)
- If Y is the ciphertext then X is some plaintext message M

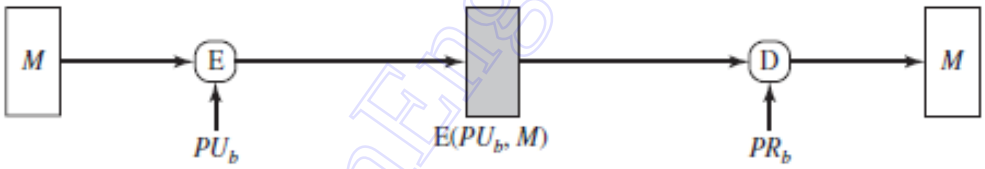


(a) Symmetric encryption: confidentiality and authentication

Public-key Encryption

(i) Public-key encryption: Confidentiality

- The source (A) uses the public key KU_b of the destination (B) to encrypt M. Because only B has the corresponding private key KR_b, only B can decrypt the message.
- This scheme provides no authentication because any opponent could also use B's public key to encrypt a message, claiming to be A.



(b) Public-key encryption: confidentiality

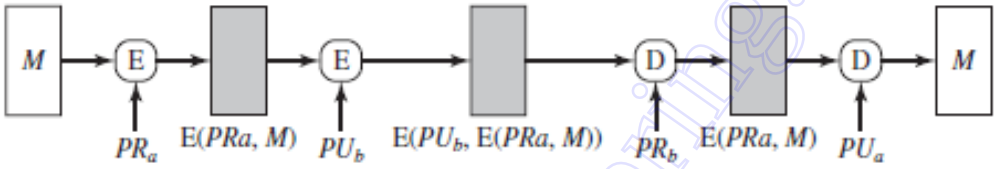
(ii) Public – key encryption: Authentication and Signature

- A uses it private key to encrypt the message, and B uses A's public key to decrypt.
- This provides authentication using the same type of reasoning as in the symmetric encryption case : -
  - The message must have come from A because A is the only party that possesses KR_a and therefore the only party with the information necessary to construct ciphertext that can be decrypted with KUa.
  - It also provides what is known as digital signature. Only A could have constructed the ciphertext because only A possesses KR_a. Not even B, the recipient, could have constructed the ciphertext.

(c) Public-key encryption: authentication and signature

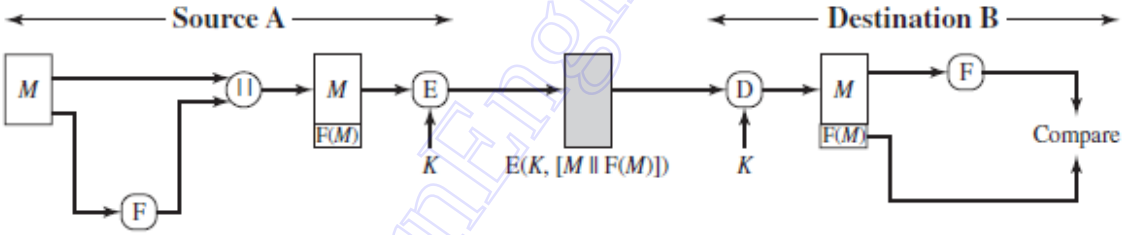
(iii) Public – key encryption: Confidentiality, Authentication and Signature

- To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality.

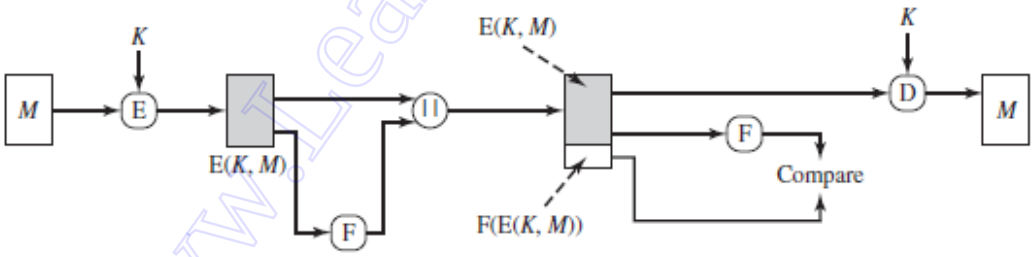


(d) Public-key encryption: confidentiality, authentication, and signature

Internal and External Error control



(a) Internal error control



(b) External error control

- A prepares a plaintext message M and then provides this input to a function F that produces an FCS (Frame Check Sequence) or Checksum.
- The FCS is appended to M and the entire block is then encrypted.
- At the Destination B decrypts the incoming block and treats the results as a message with an appended FCS.
- B applies the same function F to attempt to produce the FCS.
- If the calculated FCS is equal to the incoming FCS then the message is considered authentic.

- With internal error control, authentication is provided because an opponent would have difficulty generating ciphertext, when decrypted would have valid error control bits.
- If instead the FCS is the outer code an opponent can construct messages with valid error-control codes.
- Although the opponent cannot know what the decrypted plaintext will be, he or she can still hope to create confusion and disrupt operations.

TCP Segment:

- An error contol code added to the transmitted message serves to strengthen the authentication capability.
- Such structure is provided by the use of a communications architecture consisting of layered protocols.
- The structured message is transmitted using TCP/IP protocol architecture.
- The TCP Header:-
  - Each pair of hosts shared a unique secret key so that all exchanges between a pair of hosts used the same key.
- Then one could simply encrypt all of the datagram except the IP Header.
- If an opponent substituted some arbitrary bit pattern for the encrypted TCP segment the resulting plaintext would not a meaningful Header.
- In this case the Header includes not only a check-sum but other useful information (such as the sequence number).

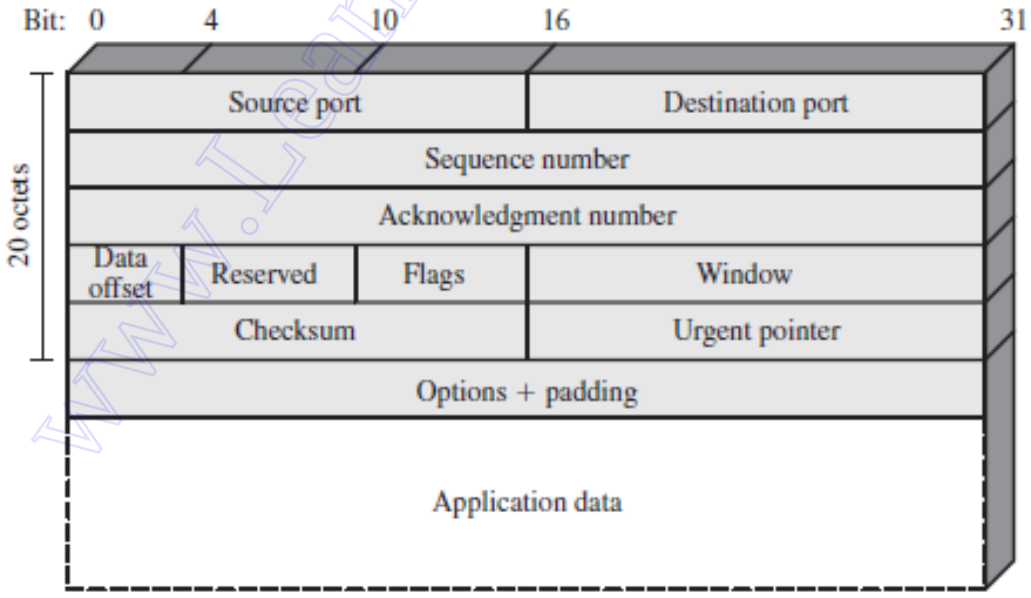


Figure: TCP Segment

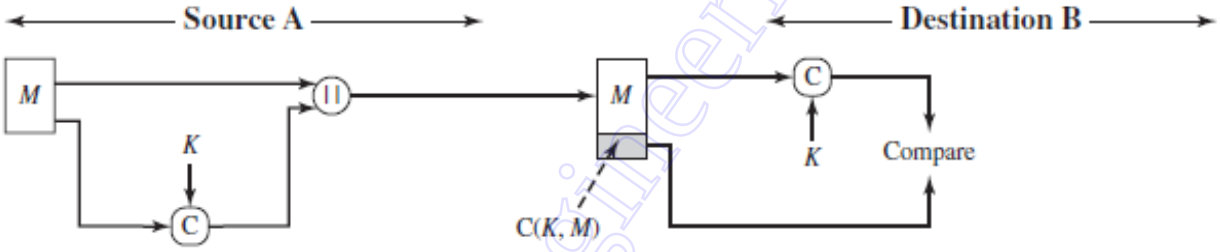
b. MESSAGE AUTHENTICATION CODE (MAC)

Message Authentication Code technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or MAC that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key *K*. When A has a message to send to B, it calculates the MAC as a function of the message and the key:

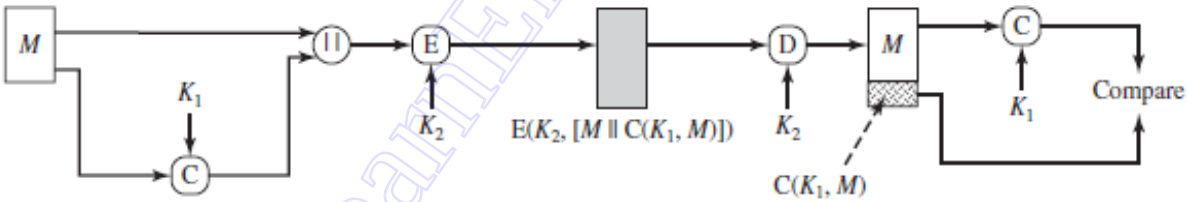
MAC = C(*K*, *M*)

where

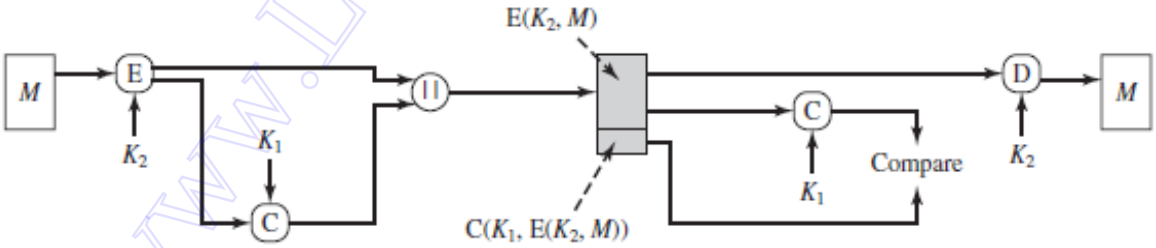
M = input message
C = MAC function
K = shared secret key
MAC = message authentication code



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

The message plus MAC are transmitted to the intended recipient. The recipient 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. If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
3. If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

The following are three situations in which a message authentication code is used.

1. There are a number of applications in which the same message is broadcast to a number of destinations. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication code. The responsible system has the secret key and performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.
2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, messages being chosen at random for checking.
3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources.

SECURITY OF MACs

We can group attacks on MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

- A brute-force attack on a MAC is a more difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs.
- To attack a hash code, given a fixed message x with n-bit hash code $h = H(x)$, a brute-force method of finding a collision is to pick a random bit string y and check if $H(y) = H(x)$.
- The attacker can do this repeatedly off line. Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the key and the tag.

The security property of a MAC algorithm can be expressed as follows.

- ✓ **Computation resistance:** Given one or more text-MAC pairs [x_i, MAC(K, x_i)], it is computationally infeasible to compute any text-MAC pair [x, MAC(K, x)] for any new input $x \neq x_i$.
- In other words, the attacker would like to come up with the valid MAC code for a given message x. There are two lines of attack possible: attack the key space and attack the MAC value.
- If an attacker can determine the MAC key, then it is possible to generate a valid MAC value for any input x.
- Suppose the key size is k bits and that the attacker has one known text-tag pair. Then the attacker can compute the n-bit tag on the known text for all possible keys. At least one key is guaranteed to produce the correct tag, namely, the valid key that was initially used to produce the known text-tag pair. This phase of the attack takes a level of effort proportional to $2k$ (that is, one operation for each of the $2k$ possible key values).
- MAC is a many-to-one mapping, there may be other keys that produce the correct value. Thus, if more than one key is found to produce the correct value, additional text-tag pairs must be tested. It can be shown that the level of effort drops off rapidly with each additional text-MAC pair and that the overall level of effort is roughly $2k$.
- An attacker can also work on the tag without attempting to recover the key. The objective is to generate a valid tag for a given message or to find a message that matches a given tag. In either case, the level of effort is comparable to that for attacking the one-way or weak collision-resistant property of a hash code, or 2^n.
- In the case of the MAC, the attack cannot be conducted off line without further input; the attacker will require chosen text-tag pairs or knowledge of the key.
- The level of effort for brute-force attack on a MAC algorithm can be expressed as min(2^k, 2^n). The assessment of strength is similar to that for symmetric encryption algorithms. It would appear reasonable to require that the key length and tag length satisfy a relationship such as min(k, n) $\geq N$, where N is perhaps in the range of 128 bits.

Cryptanalysis

- As with encryption algorithms and hash functions, cryptanalytic attacks on MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.
- The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a bruteforce attack. That is, an

ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

- There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs.

c. HASH FUNCTION

Hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M). A hash code does not use a key but is a function only of the input message. Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a **message digest** or **hash value**.

The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

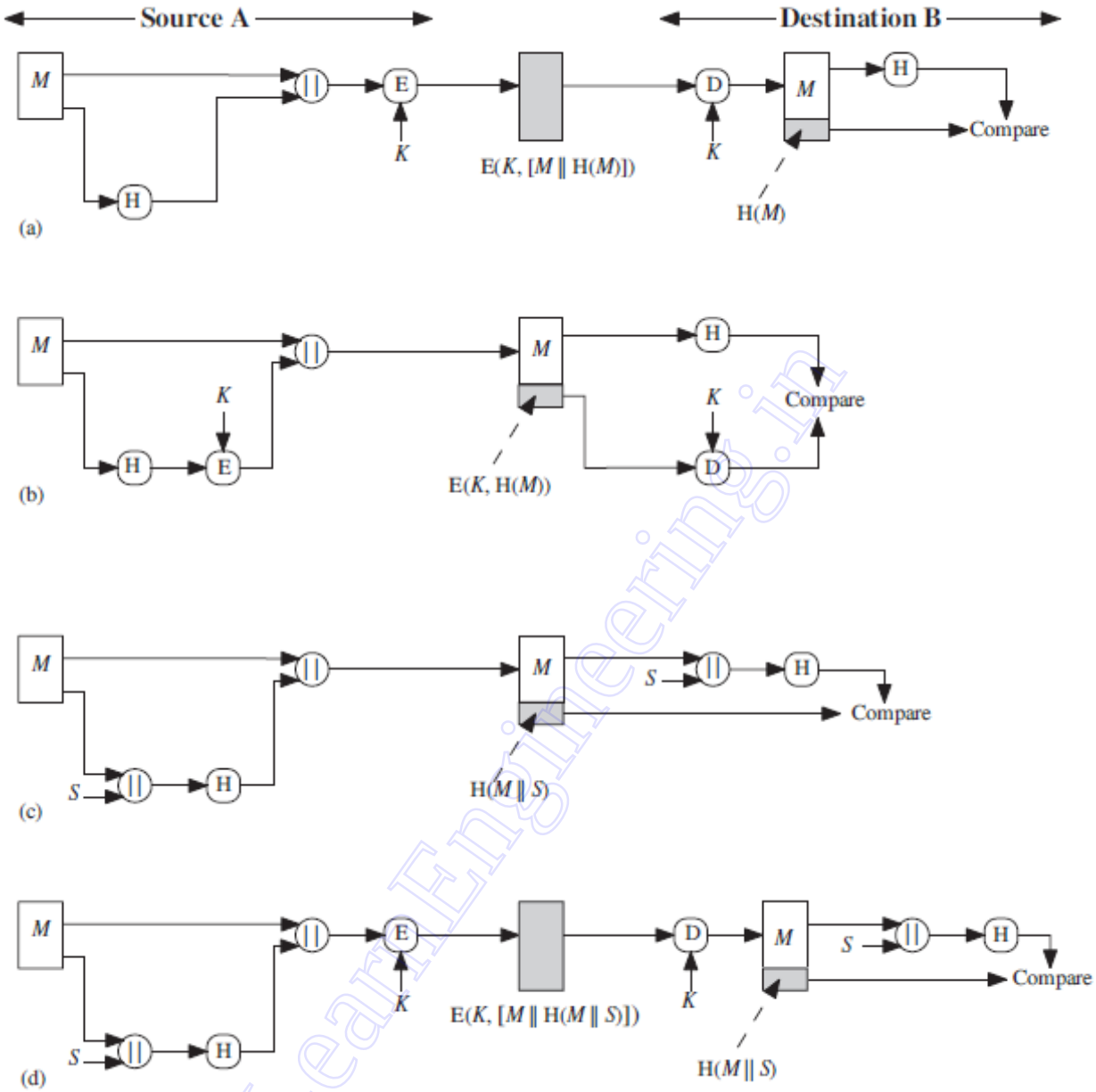


Figure: Simplified Examples of the Use of a Hash Function for Message Authentication

The above figure illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows.

a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

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

c) It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties

share a common secret value *S*. A computes the hash value over the concatenation of *M* and *S* and appends the resulting hash value to M. Because B possesses, it can recomputed the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

When confidentiality is not required, method (b) has an advantage over methods (a) and (d), which encrypts the entire message, in that less computation is required.

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

h = H(M)

where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value.

Requirements for a Hash Function

A hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to as the **one-way property**.
5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

- The first three properties are requirements for the practical application of a hash function to message authentication.
- The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value. The secret value itself is not sent; however, if the hash function is not one way, an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, the attacker obtains the message *M* and the hash code $C = H(S_{AB} \| M)$. The attacker then inverts the

hash function to obtain $S_{AB} \| M = H_1(C)$. Because the attacker now has both M and $S_{AB} \| M$, it is a trivial matter to recover S_{AB}.

- The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used.
- The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the **bit-by-bit exclusive-OR (XOR)** of every block. This can be expressed as $\boldsymbol{C_i = b_{i1} \oplus b_{i2} \oplus g \oplus b_{im}}$

where

C_i = ith bit of the hash code, 1 … i … n

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check. Each n-bit hash value is equally likely. Thus, the probability that a data error will result in an unchanged hash value is 2^{-n}. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2^{-128}, the hash function on this type of data has an effectiveness of 2^{-112}.

A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows.

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

This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input.

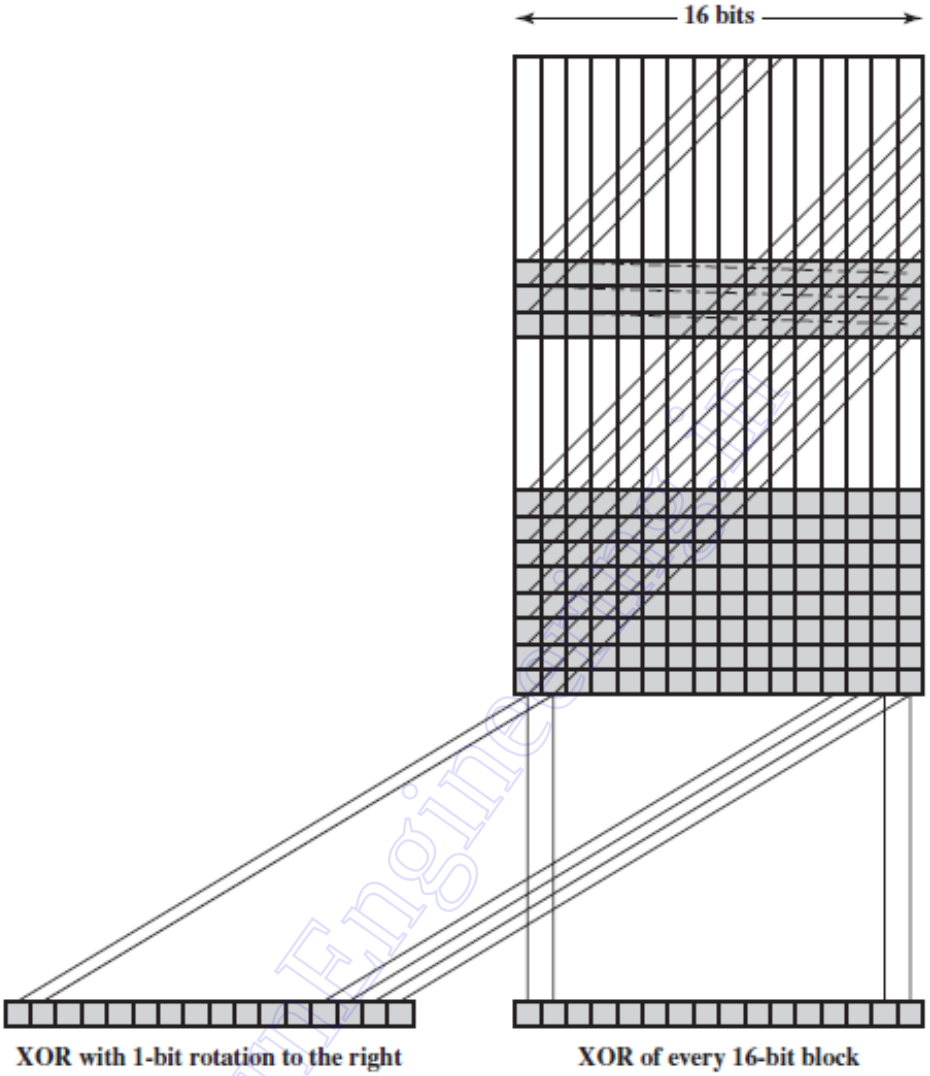


Figure: Two Simple Hash Functions

Although the second procedure provides a good measure of data integrity, it is virtually useless for data security when an encrypted hash code is used with a plaintext message. Given a message, it is an easy matter to produce a new message that yields that hash code: Simply prepare the desired alternate message and then append an *n*-bit block that forces the new message plus block to yield the desired hash code.

Although a simple XOR or rotated XOR (RXOR) is insufficient if only the hash code is encrypted. A technique originally proposed by the National Bureau of Standards used the simple XOR applied to 64-bit blocks of the message and then an encryption of the entire message that used the cipher block chaining (CBC) mode.

Given a message M consisting of a sequence of 64-bit blocks $X_1, X_2, \ldots, X_N$, define the hash code $h = \mathrm{H}(M)$ as the block-by-block XOR of all blocks and append the hash code as the final block:

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

Next, encrypt the entire message plus hash code using CBC mode to produce the encrypted message $Y_1, Y_2, \ldots, Y_{N+1}$.

For example, by the definition of CBC, we have $X_1 = IV \oplus D(K, Y_1)$

$$X_i = Y_{i-1} \oplus D(K, Y_i)$$

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

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

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

$$= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \ldots \oplus [Y_{N-1} \oplus D(K, Y_N)]$$

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

SECURITY OF HASH FUNCTION

We can group attacks on hash functions into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm. Recall from our discussion of hash functions that there are three desirable properties:

- **One-way:** For any given code h, it is computationally infeasible to find x such that $H(x) = h$.
- **Weak collision resistance:** For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
- **Strong collision resistance:** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

For a hash code of length n, the level of effort required is as follows:

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

Cryptanalyst

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits. The final block also includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult. Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.

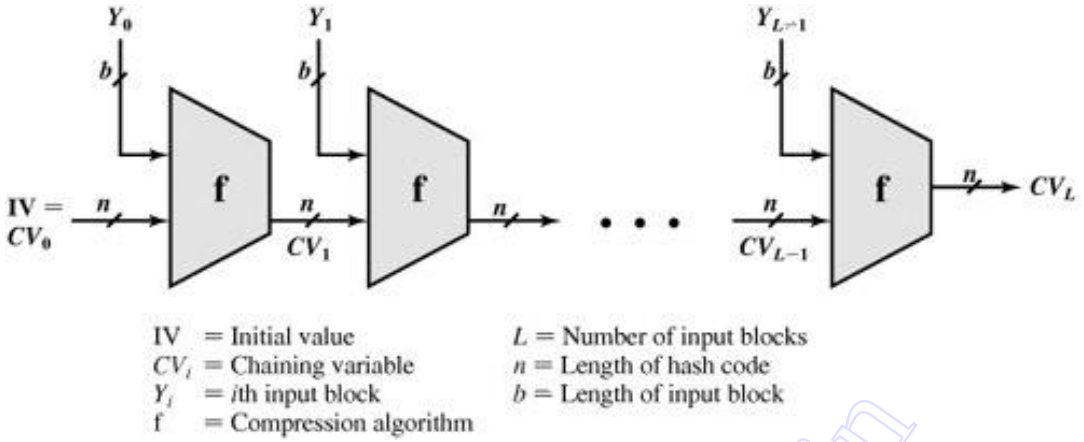


Figure: General Structure of secure hash code

The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an *n*-bit input from the previous step, called the *chaining variable*, and a *b*-bit block) and produces an *n*-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, $b > n$; hence the term *compression*. The hash function can be summarized as follows:

$$\mathbf{CV_0 = IV = Initial\ n\text{-}bit\ value}$$

$$\mathbf{CV_i = f(CV_{i1}, Y_{i1})\ \ 1 \le i \le L}$$

$$\mathbf{H(M) = CV_L}$$

The motivation for this iterative structure is that if the compression function is collision resistant, then so is the resultant iterated hash function. Therefore, the structure can be used to produce a secure hash function to operate on a message of any length. The problem of designing a secure hash function reduces to that of designing a collision-resistant compression function that operates on inputs of some fixed size.

Cryptanalysis of hash functions focuses on the internal structure of f and is based on attempts to find efficient techniques for producing collisions for a single execution of f. Once that is done, the attack must take into account the fixed value of IV. The attack on f depends on exploiting its internal structure.

Typically, as with symmetric block ciphers, f consists of a series of rounds of processing, so that the attack involves analysis of the pattern of bit changes from round to round.

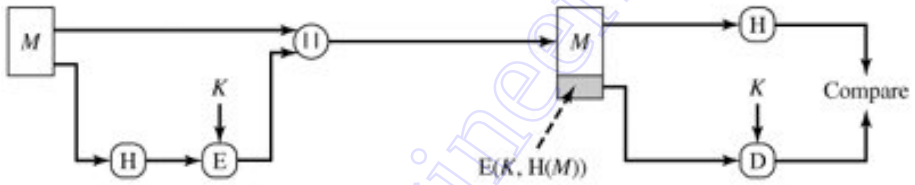
For any hash function there must exist collisions, because we are mapping a message of length at least equal to twice the block size *b* (because we must append a length field) into a hash code of length *n*, where $b \ge n$. What is required is that it is computationally infeasible to find collisions.

2. BIRTHDAY ATTACK

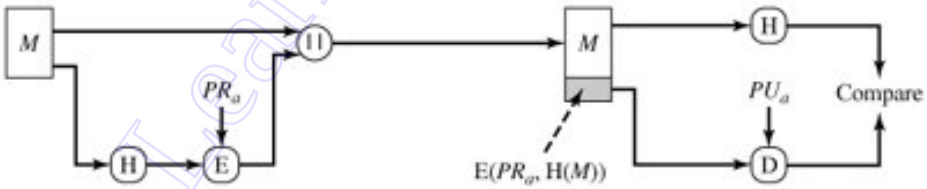
- **Illustrate about the Birthday attacks. (Nov/Dec'13)**
- **Write short notes on Birthday attack. (May/June'12)**

BIRTHDAY ATTACK

Birthday attack is used to find the collision in a cryptographic hash function. Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted message *M*, then an opponent would need to find an *M'* such that H(*M'*) = H(*M*) to substitute another message and fool the receiver. On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message.



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



In above figure, only the hash code is encrypted, using public-key encryption and using the sender's private key. As with above, this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

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

1. The source, A, is prepared to "sign" a message by appending the appropriate *m*-bit hash code and encrypting that hash code with A's private key.
2. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. The opponent prepares an equal number of

messages, all of which are variations on the fraudulent message to be substituted for the real one.

3. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.
4. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of 2^{32}.

3. MD5, SECURE HASH ALGORITHM

- **Explain about MD5 in detail. (April/May'11, April/May'10 & May/June'12)**
- **Explain Secure Hashing Algorithm (SHA) (April/May'15, Nov/Dec'13, May/June'13 & April/May'10)**
- **Explain the process of deriving eighty 64-bit words from the 1024-bits for processing of a single block and also discuss single round function in SHA-512 algorithm. Show the results of W_{16}, W_{17}, W_{18} and W_{19}. (Nov/Dec'14)**

a. MESSAGE DIGEST 5: MD5

- ✓ Developed by Ron Rivest at MIT
- ✓ Input: a message of arbitrary length
- ✓ Output: 128-bit message digest
- ✓ 32-bit word units, 512-bit blocks

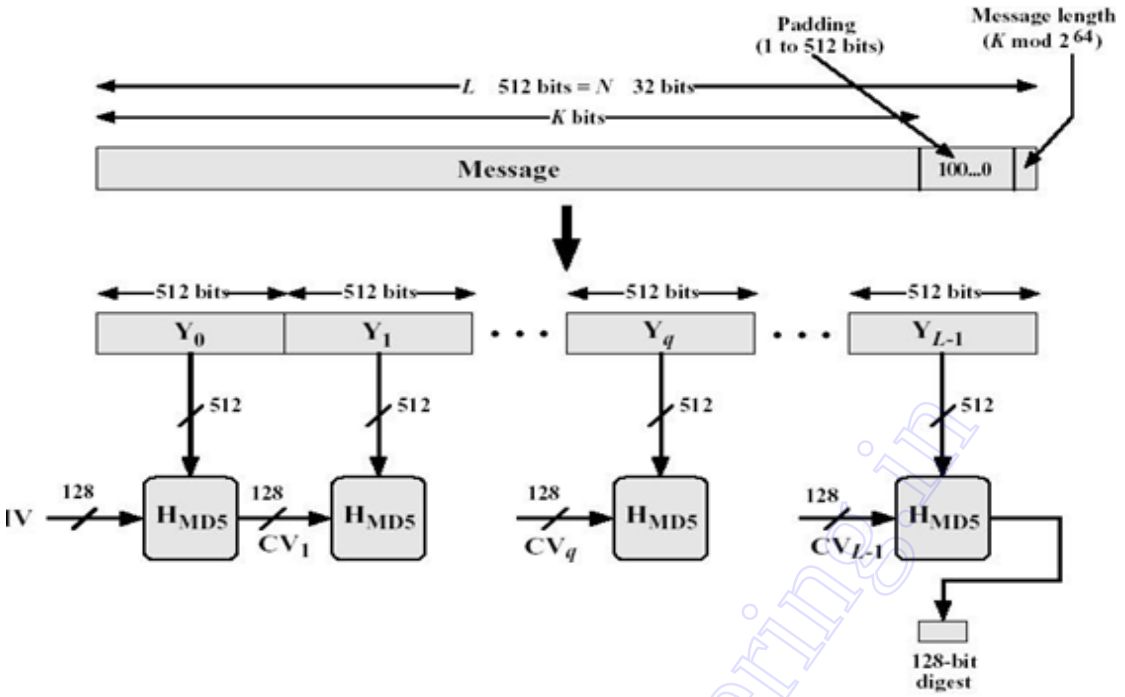


Figure: MD5 Logic

MD5 Logic:

Step 1: Append padding bits

- ✓ The message is Padded so that its bit length ≡ 448 mod 512 (i.e., the length of padded message is 64 bits less than an integer multiple of 512 bits)
- ✓ Padding is always added, even if the message is already of the desired length (1 to 512 bits)
- ✓ Padding bits: 1000….0 (a single 1-bit followed by the necessary number of 0-bits)

Step 2: Append length

- ✓ A 64-bit length: contains the length of the original message modulo 264
- ✓ The expanded message is $Y_0, Y_1, \ldots, Y_{L-1}$; the total length is L × 512 bits
- ✓ The expanded message can be thought of as a multiple of 16 32-bit words
- ✓ Let M[0 … N-1] denote the word of the resulting message, where N = L × 16

Step 3: Initialize MD buffer

- ✓ 128-bit buffer (four 32-bit registers A,B,C,D) is used to hold intermediate and final results of the hash function
- ✓ A,B,C,D are initialized to the following values

A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476

- ✓ Stored in *little-endian* format (least significant byte of a word in the low-address byte position)
  - E.g. word A : 01 23 45 67 (low address … high address)
  - word B : 89 AB CD EF
  - word C : FE DC BA 98
  - word D : 76 54 32 10

Step 4: Process message in 512-bit (16-word) blocks

- ✓ Heart of the algorithm called a *compression function* Consists of 4 rounds
- ✓ The 4 rounds have a similar structure, but each uses a different *primitive logical functions*, referred to as F, G, H, and I
- ✓ Each round takes as input the current 512-bit block (Yq), 128-bit buffer value ABCD and updates the contents of the buffer
- ✓ Each round also uses the table T[1 … 64], constructed from the sine function;
  - T[i] = 232 × abs(sin(i))
- ✓ The output of 4th round is added to the CVq to produce CVq+1

Step 5: Output

- ✓ After all L 512-bit blocks have been processed, the output from the Lth stage is the 128-bit message digest

$$\mathbf{CV_0 = IV}$$

$$\mathbf{CVq+1 = SUM32(CVq, RFI[Yq, RFH[Yq, RFG[Yq, RFF[Yq, CVq]]])}$$

$$\mathbf{MD = CV_L}$$

Where IV = initial value of the ABCD buffer, defined in step 3

Yq = the qth 512-bit block of the message

L = the number of blocks in the message (including padding and length fields) CVq = chaining variable processed with the qth block of the message

RFx = round function using primitive logical function x

MD = final message digest value

SUM32 = addition modulo 232 performed separately on each word

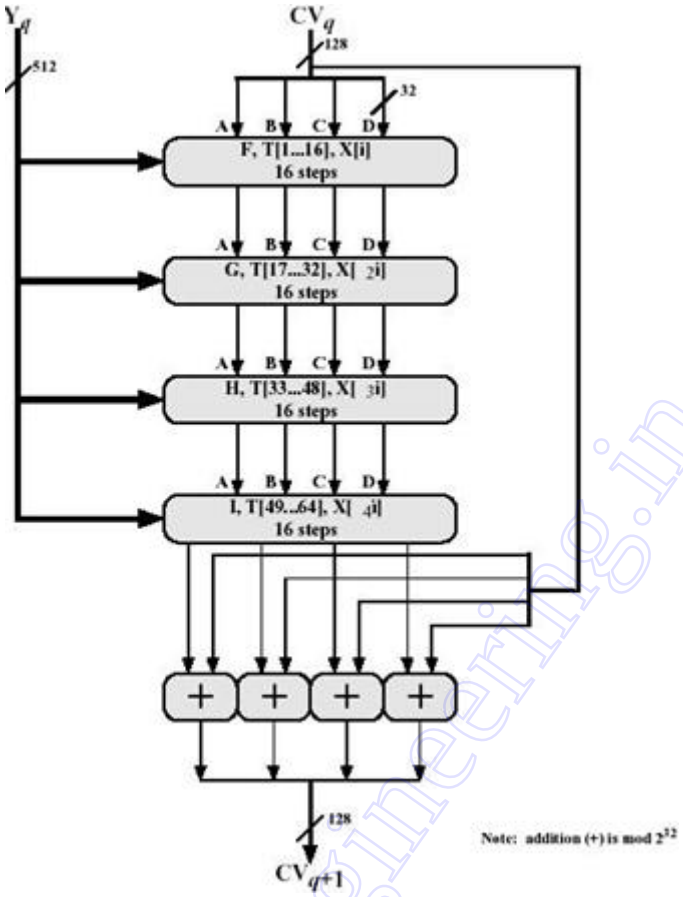


Figure: MD5 processing of a single 512-bit block (MD5 compression function)

MD5 Compression Function:

- ✓ Each round consists of a sequence of 16 steps operating on the buffer ABCD
- ✓ Each step is of the form

$$a \leftarrow b + ((a + g(b, c, d) + X[k] + T[i] <<< s)$$

where

a,b,c,d = the 4 words of the buffer, in a specified order that varies across steps

g = one of the primitive functions F, G, H, I

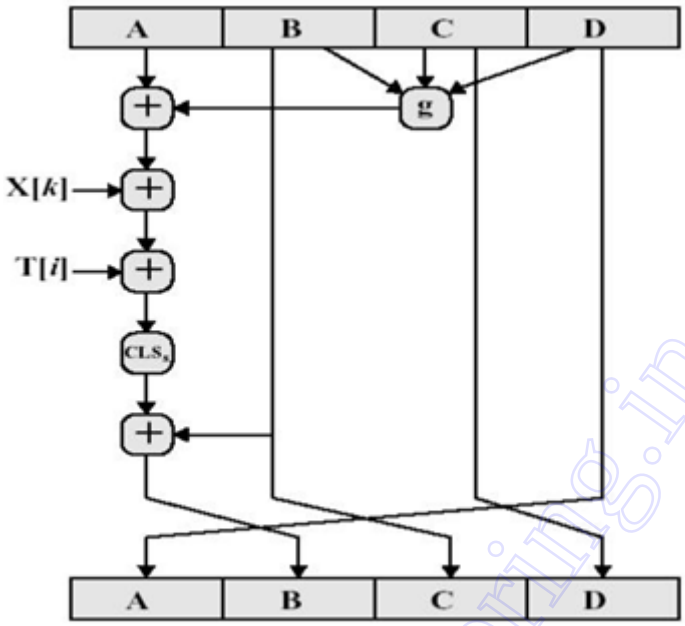
<<s = circular left shift (rotation) of the 32-bit arguments by s bits

X[k] = M[q × 16 + k] = the kth 32-bit word in the qth 512-bit block of the message

T[i] = the ith 32-bit word in table T

+ = addition modulo 232

MD5 Operation



- ✓ One of the 4 primitive logical functions is used in each 4 rounds of the algorithm
- ✓ Each primitive function takes three 32-bit words as input and produces a 32-bit word output
- ✓ Each function performs a set of bitwise logical operations

Round	Primitive function g	g(b, c, d)
1	F(b, c, d)	(b ∧ c) ∨ (b' ∧ d)
2	G(b, c, d)	(b ∧ d) ∨ (c ∧ d')
3	H(b, c, d)	b ⊕ c ⊕ d
4	I(b c, d)	c ⊕ (b ∨ d')

TRUTH TABLE						
b	**C**	**d**	**F**	**G**	**H**	**I**
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0

- ✓ The array of 32-bit words X[0..15] holds the value of current 512-bit input block being processed

- ✓ Within a round, each of the 16 words of X[i] is used exactly once, during one step
  - The order in which these words is used varies from round to round
  - In the first round, the words are used in their original order
  - For rounds 2 through 4, the following permutations are used
    - » ρ2(i) = (1 + 5i) mod 16
    - » ρ3(i) = (5 + 3i) mod 16
    - » ρ4(I) = 7i mod 16

b. SECURE HASH ALGORITHM

- Developed by NIST (National Institute of Standards and Technology)
  - Published as a FIPS 180 in 1993
  - A revised version is issued as FIPS 180-1 IN 1995
  - Generally referred to as SHA-1
- SHA is based on the hash function MD4 and its design closely models MD4.
- SHA- 1 produces a hash value of 160 bits.
- Revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384 and 512 bits, known as SHA-256, SHA-384 and SHA-512.

SHA-512 Logic:

The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

Step 1: Append padding bits

- The message is Padded so that its bit length is congruent to 896 modulo 1024 [length K ≡ 896 mod 1024]
- Padding is always added, even if the message is already of the desired length.
- Thus, the number of padding bits is in the range of 1 to 1024.
- The padding consists of a single 1-bit followed by the necessary number of 0-bits.

Step 2: Append length

- A block of 128-bits is appended to the message.
- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure, the expanded message is represented as the sequence of

1024-bit blocks M_1, M_2,.., M_n, so that the total length of the expanded message is N x 1024 bits.

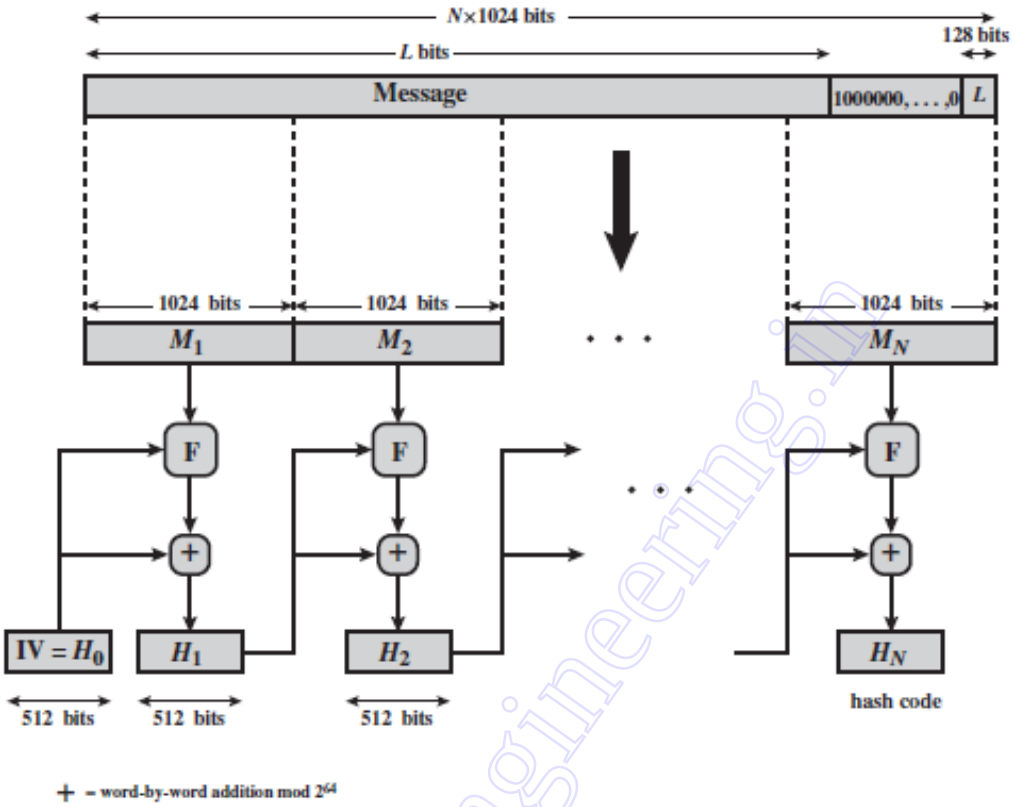


Figure: Message Digest Generation using SHA-512

Step 3: Initialize hash buffer

- A 512-bit buffer is used to hold intermediate and final results of the hash function.
- The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g and h).
- These registers are initialized to the following 64-bit integers (hexadecimal values):

```
a = 6A09E667F3BCC908    e = 510E527FADE682D1
b = BB67AE8584CAA73B    f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B    g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1    h = 5BE0CD19137E2179
```

- These values are stored in **big-endian** format, which is the most significant byte of a word in the low-address (leftmost) byte position.
- These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4: Process message in 1024-bit (128-word) blocks

- The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in above figure.
- Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer.
- At input to the first round, the buffer has the value of the intermediate hash value, H_{i-1}.
- Each round *t* makes use of a 64-bit value W_t, derived from the current 1024-bit block being processed (M_i).
- Each round also makes use of an additive constant K_t, where $0 \leq t \leq 79$ indicates one of the 80 rounds.
- The output of the eightieth round is added to the input to the first round (H_{i-1}) to produce H_i. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1}, using addition modulo 2^{64}.

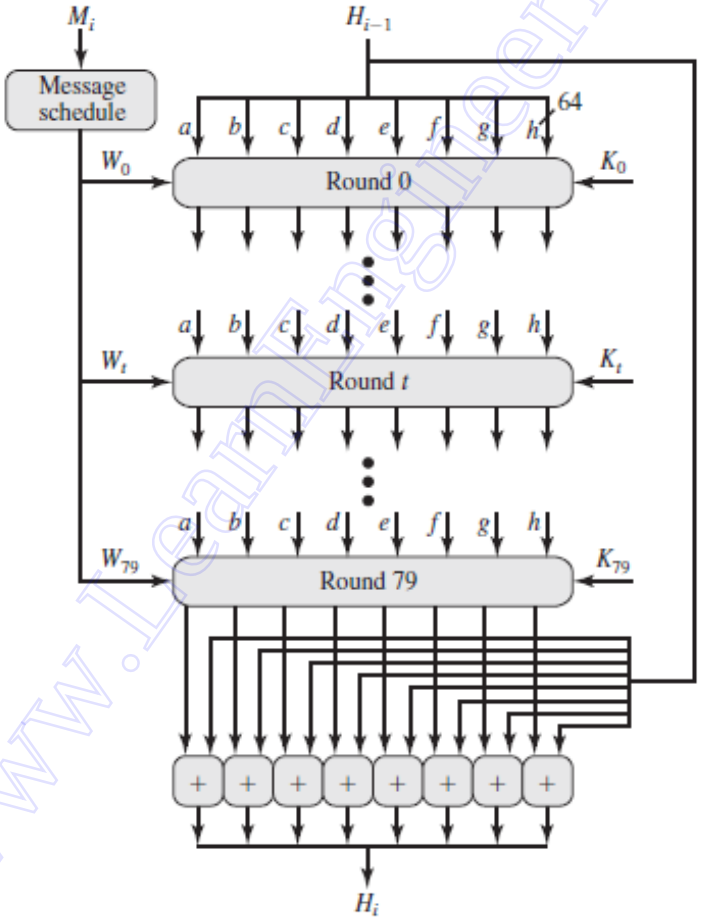


Figure: SHA-512 processing of a single 1024-bit block

Step 5: Output

After all *N* 1024-bit blocks have been processed, the output from the N^{th} stage is the 512-bit message digest.

$\mathbf{H_0 = IV}$

$\mathbf{H_i = SUM_{64}(H_{i-1}, abcdefgh_i)}$

$\mathbf{MD = H_N}$

where,

IV = initial value of the abcdefgh buffer, defined in step 3.

$\mathbf{abcdefgh_i}$ = the output of the last round of processing of the i^{th} message block.

N = the number of blocks in the message (including padding and length fields).

$\mathbf{SUM_{64}}$ = Addition modulo 2^{64} performed separately on each word of the pair of inputs.

MD = final message digest value.

SHA-512 Round Functions:

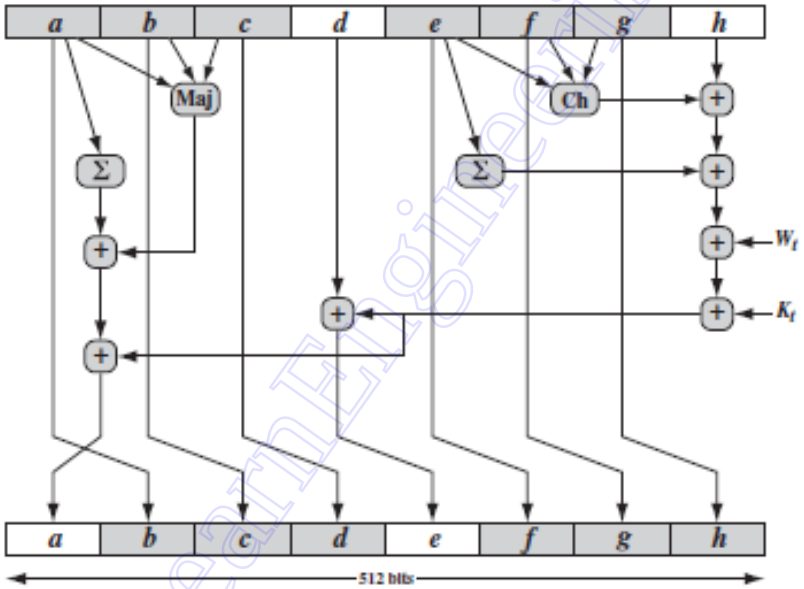


Figure: Elementary SHA-512 operation (single round)

Each round is defined by the following set of equations:

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

where

t = step number; $0 \le t \le 79$

$\text{Ch}(e, f, g)$ = $(e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$
the conditional function: If e then f else g

$\text{Maj}(a, b, c)$ = $(a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

$\left(\sum_0^{512} a\right)$ = $\text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right)$ = $\text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

W_t = a 64-bit word derived from the current 1024-bit input block

K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

It remains to indicate how the 64-bit word values W_t are derived from the 1024-bit message. The first 16 values of W_t are taken directly from the 16 words of the current block. The remaining values are defined as:

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$

$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$\text{SHR}^n(x)$ = left shift of the 64-bit argument x by n bits with padding by zeros on the right

$+$ = addition modulo 2^{64}

Thus, in the first 16 steps of processing, the value of W_t is equal to the corresponding word in the message block. For the remaining 64 steps, the value of W_t

consists of the circular left shift by one bit of the XOR of four of the preceding values of W_t, with two of those values subjected to shift and rotate operations.

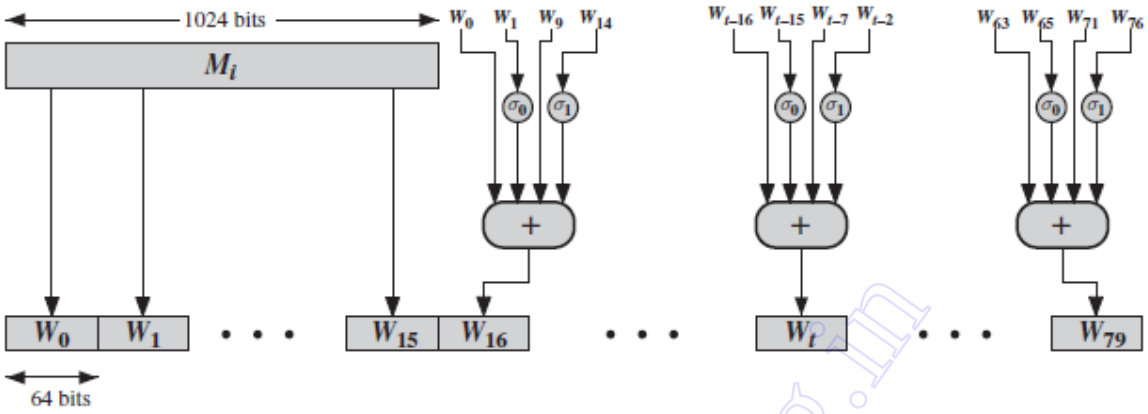


Fig. Creation of 80-word Input Sequence

4. HMAC and CMAC

- **Explain in detail about HMAC and CMAC.**
- **Discuss about the objectives of HMAC and its security features. (May/June'07)**

a. MACs BASED ON HASH FUNCTIONS (HMAC)

- ✓ HMAC has been issued as RFC 2104, has been chosen as the mandatory-to-implement MAC for IP security, and is used in other Internet protocols, such as SSL.
- ✓ HMAC has also been issued as a NIST standard (FIPS 198).

HMAC Design Objectives:

- To use, without modifications, in available hash functions. In particular, to use hash functions that perform well in software and for which code is freely and widely available.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

HMAC Algorithm:

HMAC defines the following terms.

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
IV = initial value input to hash function
M = message input to HMAC (including the padding specified in the embedded hash function)
Y_i = i th block of M, $0 \leq i \leq (L - 1)$
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; recommended length is $\geq n$; if key length is greater than b, the key is input to the hash function to produce an n-bit key
K^+ = K padded with zeros on the left so that the result is b bits in length
ipad = 00110110 (36 in hexadecimal) repeated $b/8$ times
opad = 01011100 (5C in hexadecimal) repeated $b/8$ times

Then HMAC can be expressed as

$$\mathbf{HMAC}(K, M) = \mathbf{H}[(K^+ \oplus \mathbf{opad}) \| \mathbf{H}[(K^+ \oplus \mathbf{ipad}) \| M]]$$

The algorithm is as follows:

1. Append zeros to the left end of K to create a b-bit string K^+ (e.g., if K is of length 160 bits and b = 512, then K will be appended with 44 zeroes).
2. XOR (bitwise exclusive-OR) K^+ with ipad to produce the b-bit block S_i.
3. Append M to S_i.
4. Apply H to the stream generated in step 3.
5. XOR K^+ with opad to produce the b-bit block S_o.
6. Append the hash result from step 4 to S_o.
7. Apply H to the stream generated in step 6 and output the result.

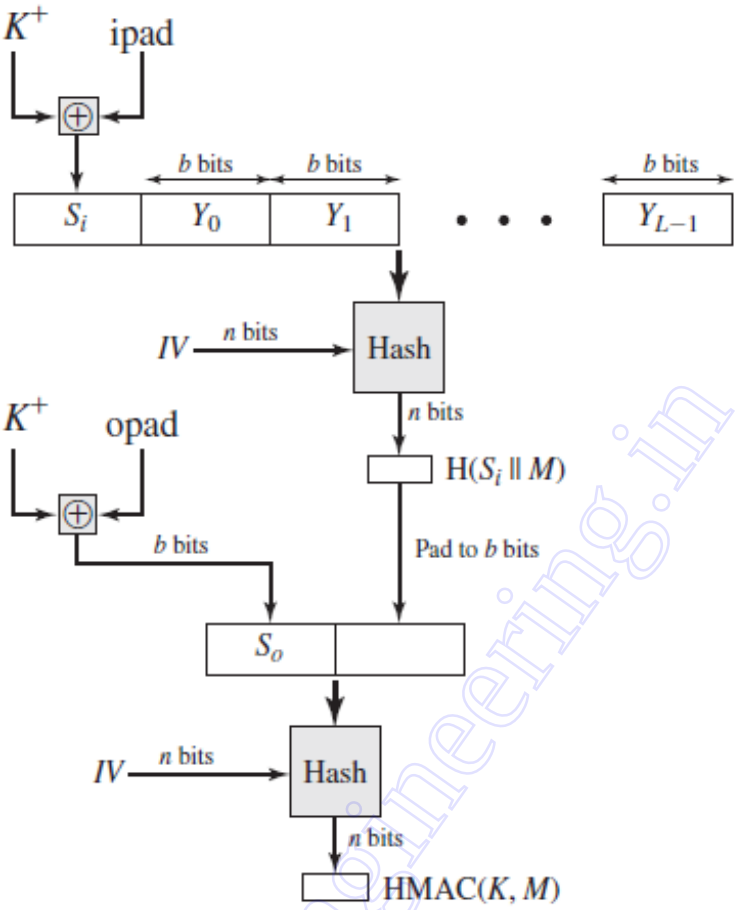


Figure: HMAC Structure

The XOR with ipad results in flipping one-half of the bits of K. Similarly, the XOR with opad results in flipping one-half of the bits of K, using a different set of bits. In effect, by passing S_i and S_o through the compression function of the hash algorithm, we have pseudorandomly 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_i, S_o, and the block produced from the inner hash). A more efficient, two quantities are precomputed:

$$\mathbf{f}(IV, (K^+ \oplus \mathbf{ipad}))$$
$$\mathbf{f}(IV, (K^+ \quad \mathbf{opad}))$$

where f(cv, block) 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. These quantities only need to be computed initially and every time the key changes.

In effect, 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 more efficient implementation is especially worthwhile if most of the messages for which a MAC is computed are short.

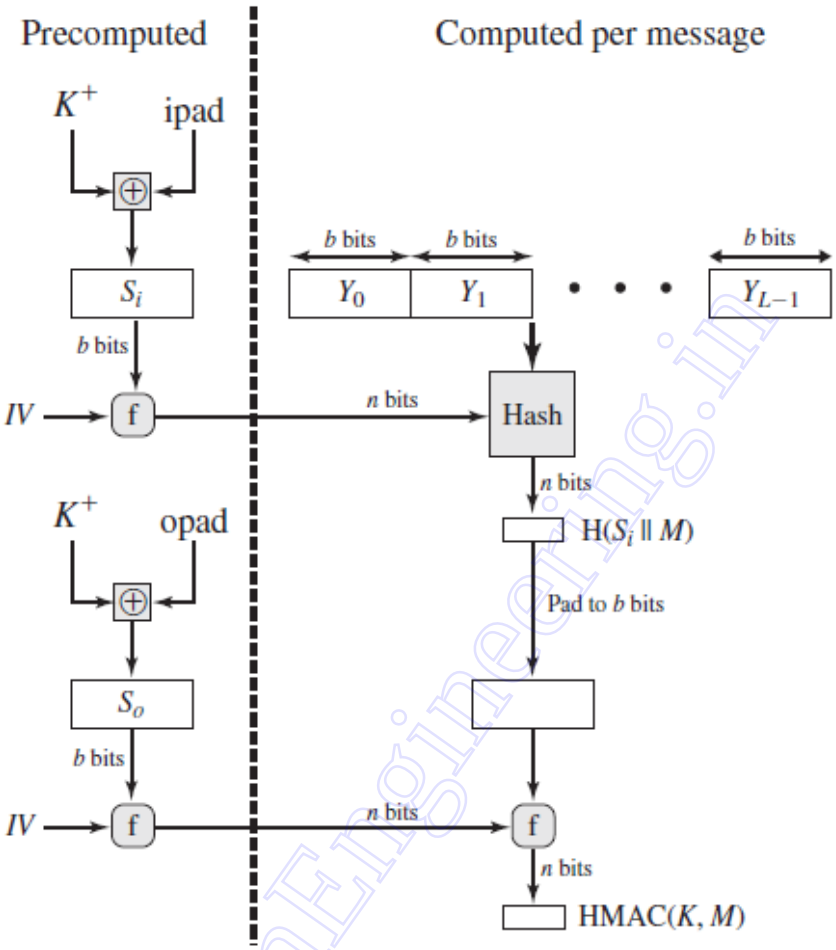


Figure: Efficient implementation of HMAC

Security of HMAC :

The security of any MAC function based on an embedded hash function. 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-tag pairs created with the same key. For a given level of effort (time, message–tag pairs) on messages generated by a legitimate user and seen by the attacker, the probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function.

- ✓ The attacker is able to compute an output of the compression function even with an *IV* that is random, secret, and unknown to the attacker.
- ✓ The attacker finds collisions in the hash function even when the *IV* is random and secret.

In the first attack, we can view the compression function as equivalent to the hash function applied to a message consisting of a single *b*-bit block. For this attack,

the *IV* of the hash function is replaced by a secret, random value of *n* bits. An attack on this hash function requires either a brute-force attack on the key, which is a level of effort on the order of 2^n, or a birthday attack.

In the second attack, the attacker is looking for two messages *M* and *M'* that produce the same hash: H(*M*) = H(*M'*). This is the birthday attack. This requires a level of effort of $2^{n/2}$ for a hash length of *n*.

b. CIPHER-BASED MESSAGE AUTHENTICATION CODE (CMAC)

CMAC 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.

When the message is an integer multiple *n* of the cipher block length *b*. For AES, *b* = 128, and for triple DES, *b* = 64. The message is divided into *n* blocks (M_1, M_2, …, M_n). The algorithm makes use of a *k*-bit encryption key *K* and a *b*-bit constant, K_1. For AES, the key size *k* is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows.

$$C_1 = \mathbf{E}(K, M_1)$$
$$C_2 = \mathbf{E}(K, [M_2 \quad C_1])$$
$$C_3 = \mathbf{E}(K, [M_3 \quad C_2])$$
$$\vdots$$
$$C_n = \mathbf{E}(K, [M_n \quad C_{n-1} \quad K_1])$$
$$T = \mathbf{MSB}_{Tlen}(C_n)$$

where

T = message authentication code, also referred to as the tag
Tlen = bit length of T
MSBs(*X*) = the *s* leftmost bits of the bit string *X*

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length *b*. The CMAC operation then proceeds as before, except that a different *b*-bit key K_2 is used instead of K_1.

The two *b*-bit keys are derived from the *k*-bit encryption key as follows.

$$L = \mathbf{E}(K, 0^b)$$
$$K_1 = L \,.\, x$$
$$K_2 = L \,.\, x^2 = (L \,.\, x) \,.\, x$$

where multiplication (.) is done in the finite field GF(2^b) and *x* and x^2 are first and second-order polynomials that are elements of GF(2^b).

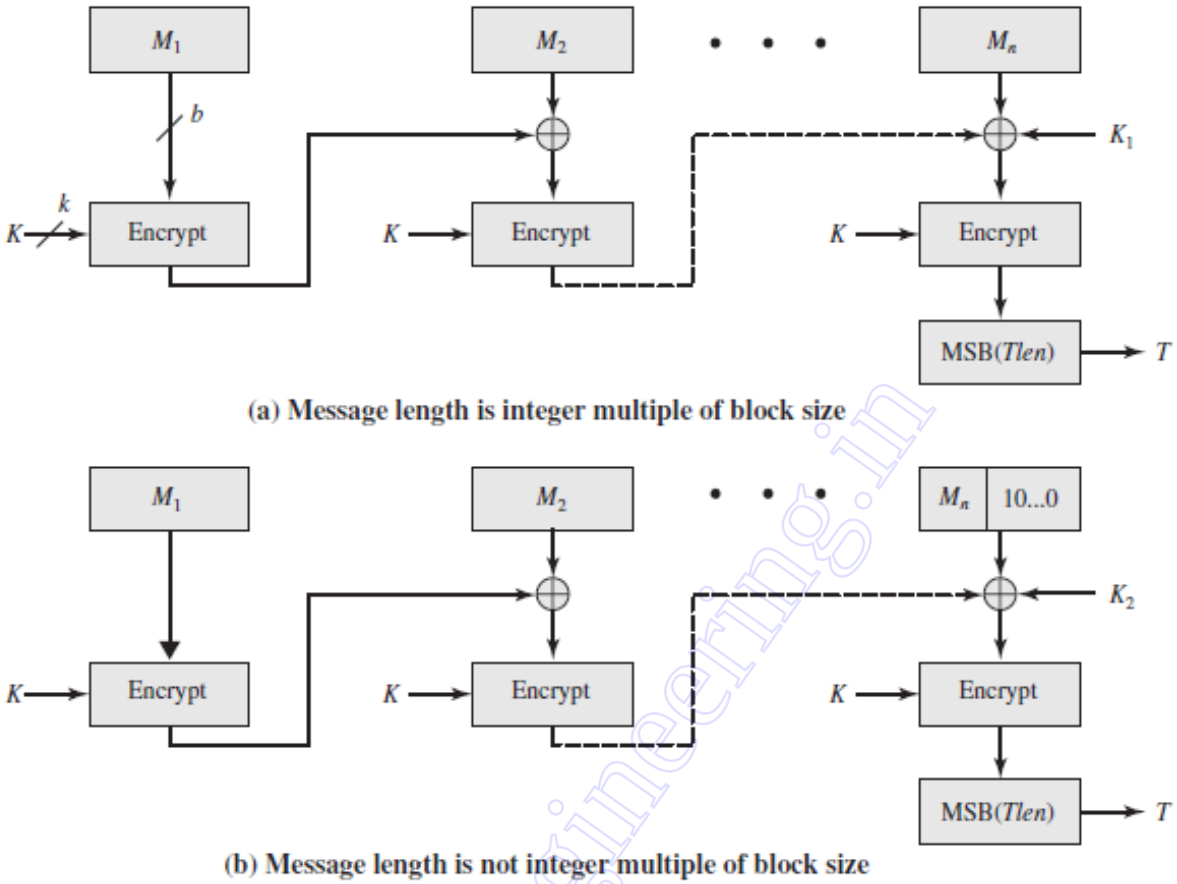


Figure: Cipher-Based Message Authentication Code

To generate K_1 and K_2, the block cipher is applied to the block that consists entirely of 0 bits. The first subkey is derived from the resulting ciphertext by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size. The second subkey is derived in the same manner from the first subkey.

5. AUTHENTICATION PROTOCOLS

❖ **Explain in details about authentication protocols.**

AUTHENTICATION PROTOCOLS:

Two general areas are

- Mutual authentication and
- One-way authentication

a. Mutual Authentication

- ✓ An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.
- ✓ Central to the problem of authenticated key exchange are two issues:
  - Confidentiality &
  - Timeliness
- ✓ To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form.
- ✓ At minimum a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

The following are examples of replay attacks:

- ✓ **Simple replay:** The opponent simply copies a message and replays it later.
- ✓ **Repetition that can be logged:** An opponent can replay a time stamped message within the valid time window.
- ✓ **Repetition that cannot be detected:** This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
- ✓ **Backward replay without modification:** This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

General approaches:

- **Timestamps:**
  - Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time.
  - This approach requires that clocks among the various participants be synchronized.
- **Challenge/response:**
  - Party A, expecting a fresh message from B, first sends B a nonce(challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

Symmetric Encryption Approaches:

- A two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment. In general, this strategy involves the use of a trusted key distribution center (KDC).
- Each party in the network shares a secret key, known as a master key, with the KDC.
- The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.

1. A ⟶ KDC: $ID_A||ID_B||N_1$
2. KDC ⟶ A: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
3. A ⟶ B: $E(K_b, [K_s||ID_A])$
4. A ⟶ A: $E(K_s, N_2)$
5. A ⟶ B: $E(K_s, f(N_2))$

- Secret keys K_a and K_b are shared between A and the KDC and B and the KDC, respectively. The purpose of the protocol is to distribute securely a session key K, to A and B.
- Suppose that an opponent, X, has been able to compromise an old session key. Her proposal assumes that the master keys, K_a and K_b, are secure, and it consists of the following steps:

1. A ⟶ KDC: $ID_A||ID_B$
2. KDC ⟶ A: $E(K_a, [K_s||ID_B||T||E(K_b, [K_s||ID_A||T])])$
3. A ⟶ B: $E(K_b, [K_s||ID_A||T])$
4. B ⟶ A: $E(K_s, N_1)$
5. A ⟶ B: $E(K_s, f(N_1))$

- T is a timestamp that assures A and B that the session key has only just been generated. Thus, both A and B know that the key distribution is a fresh exchange. A and B can verify timeliness by checking that

$$|\text{Clock } T| < \Delta t_1 + \Delta t_2$$

where $\Delta t1$ is the estimated normal discrepancy between the KDC's clock and the local clock (at A or B) and $\Delta t2$ is the expected network delay time. Each node can set its clock against some standard reference source. Because the timestamp *T* is encrypted

using the secure master keys, an opponent, even with knowledge of an old session key, cannot succeed because a replay of step 3 will be detected by B as untimely.

1. A ⟶ B: $ID_A||N_a$
2. B ⟶ KDC: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])$
3. KDC ⟶ A: $E(K_a, [ID_B||N_a||K_s||T_b])||E(K_b,[ID_A||K_s||T_b])||N_b$
4. A ⟶ B: $E(K_b, [ID_A||K_s||T_b])||E(K_s, N_b)$

1. A initiates the authentication exchange by generating a nonce, N_a, and sending that plus its identifier to B in plaintext. This nonce will be returned to A in an encrypted message that includes the session key, assuring A of its timeliness.
2. B alerts the KDC that a session key is needed. Its message to the KDC includes its identifier and a nonce, N_b This nonce will be returned to B in an encrypted message that includes the session key, assuring B of its timeliness. B's message to the KDC also includes a block encrypted with the secret key shared by B and the KDC. This block is used to instruct the KDC to issue credentials to A; the block specifies the intended recipient of the credentials, a suggested expiration time for the credentials, and the nonce received from A.
3. The KDC passes on to A B's nonce and a block encrypted with the secret key that B shares with the KDC. The block serves as a "ticket" that can be used by A for subsequent authentications, as will be seen. The KDC also sends to A a block encrypted with the secret key shared by A and the KDC. This block verifies that B has received A's initial message (ID_B) and that this is a timely message and not a replay (N_a) and it provides A with a session key (K_s) and the time limit on its use (T_b).
4. A transmits the ticket to B, together with the B's nonce, the latter encrypted with the session key. The ticket provides B with the secret key that is used to decrypt $E(K_s, N_b)$ to recover the nonce. The fact that B's nonce is encrypted with the session key authenticates that the message came from A and is not a replay.

A desires a new session with **B**. The following protocol ensues:

1. A ⟶ B: $E(K_b, [ID_A||K_s||T_b])||N'_a$
2. B ⟶ A: $N'_b||E(K_s, N'_a)$
3. A ⟶ B: $E(K_s, N'_b)$

When B receives the message in step 1, it verifies that the ticket has not expired. The newly generated nonces *N'a* and *N'b* assure each party that there is no replay attack.

Public key encryption approaches

This protocol assumes that each of the two parties is in possession of the current public key of the other.

1. A ⟶ AS: $ID_A||ID_B$
2. AS ⟶ A: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])$
3. A ⟶ B: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])$ $||E(PU_b, E(PR_a, [K_s||T]))$

In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret key distribution. Rather, the AS provides public-key certificates.

The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS. The timestamps protect against replays of compromised keys.

1. A ⟶ KDC: $ID_A||ID_B$
2. KDC ⟶ A: $E(PR_{auth}, [ID_B||PU_b])$
3. A ⟶ B: $E(PU_b, [N_a||ID_A])$
4. B ⟶ KDC: $ID_A||ID_B||E(PU_{auth}, N_a)$
5. KDC ⟶ B: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_B]))$
6. B ⟶ A: $E(PU_a, E(PR_{auth}, [(N_a||K_s||ID_B)||N_b]))$
7. A ⟶ B: $E(K_s, N_b)$

This seems to be secure protocol that takes into account the various attacks. The authors themselves spotted a flaw and submitted a revised version of the algorithm is as follows.

1. A ⟶ KDC: $ID_A||ID_B$

2. KDC ⟶ A: $E(PR_{auth}, [ID_B||PU_b])$
3. A ⟶ B: $E(PU_b, [N_a||ID_A])$
4. B ⟶ KDC: $ID_A||ID_B||E(PU_{auth}, N_a)$
5. KDC ⟶ B: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_A||ID_B])$
6. B ⟶ A: $E(PU_a, E(PR_{auth}, [(N_a||K_s||ID_A||ID_B)||N_b]))$
7. A ⟶ B: $E(K_s, N_b)$

b. One way Authentication

Symmetric Encryption Approach

This scheme requires the sender to issue a request to the intended recipient, await a response that includes a session key, and only then send the message. The KDC strategy is a candidate for encrypted electronic mail. Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated.

For a message with content *M*, the sequence is as follows:

1. A ⟶ KDC: $ID_A||ID_B||N_1$
2. KDC ⟶ A: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
3. A ⟶ B: $E(K_b, [K_s||ID_A])||E(K_s, M)$

This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A.

Public Key encryption Approaches

This approach requires that either the sender know the recipient's public key (confidentiality) or the recipient know the sender's public key (authentication) or both (confidentiality plus authentication). In addition, the public-key algorithm must be applied once or twice to what may be a long message.

If confidentiality is the primary concern, then the following may be more efficient:

$$A \longrightarrow B: E(PU_b, K_s)||E(K_s, M)$$

If authentication is the primary concern, then a digital signature may suffice.

$$A \longrightarrow B: M||E(PR_a, H(M))$$

This method guarantees that A cannot later deny having sent the message. However, this technique is open to another kind of fraud.

Both the message and signature can be encrypted with the recipient's public key:

$$A \longrightarrow B: E(PU_b, [M||E(PR_a, H(M))])$$

The latter two schemes require that B know A's public key and be convinced that it is timely. An effective way to provide this assurance is the digital certificate.

$$A \longrightarrow B: M||E(PR_a, H(M))||E(PR_{as}, [T||ID_A||PU_a])$$

In addition to the message, A sends B the signature, encrypted with A's private key, and A's certificate, encrypted with the private key of the authentication server. The recipient of the message first uses the certificate to obtain the sender's public key and verify that it is authentic and then uses the public key to verify the message itself. If confidentiality is required, then the entire message can be encrypted with B's public key. Alternatively, the entire message can be encrypted with a one-time secret key; the secret key is also transmitted, encrypted with B's public key.

6. DIGITAL SIGNATURE ALGORITHM

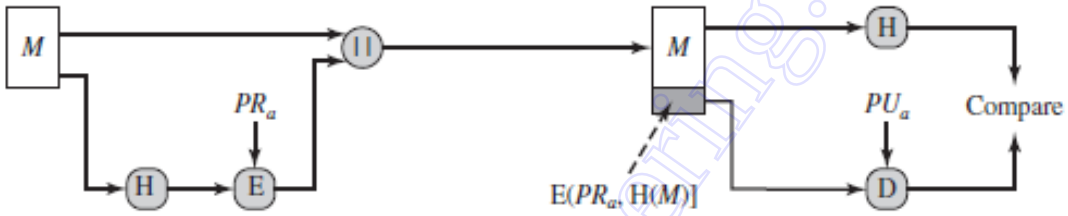
- **Explain Digital Signature Standard. (May/June'14)**
- **Give the details of digital signature algorithm. (May/June'07)**
- **With a neat sketch, explain signing and verifying functions of DSA. (May/June'12)**

NIST DIGITAL SIGNATURE ALGORITHM

- The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Algorithm (DSA).
- The DSA makes use of the Secure Hash Algorithm (SHA).
- The DSA was originally proposed in 1991 and revised in 1993, 1996 and then 2000 an expanded version of the standard was issued as FIPS 186-2, subsequently updated to FIPS 186-3 in 2009.
- The DSA uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

The RSA Approach:

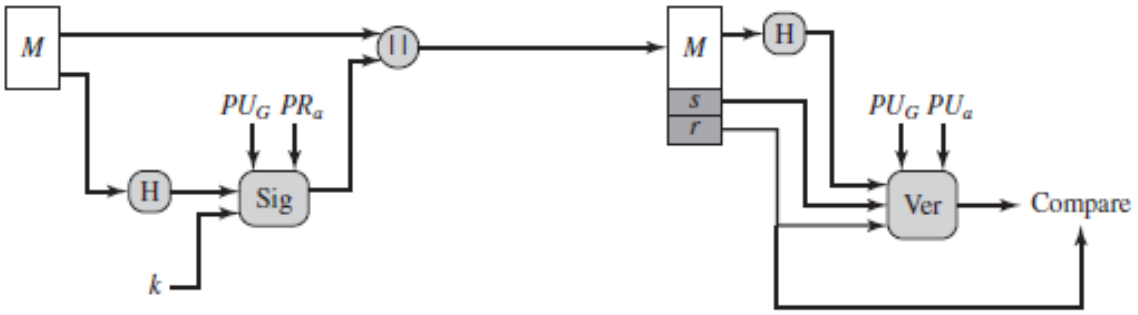
- In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length.
- This hash code is then encrypted using the sender's private key to form the signature.
- Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code.
- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.



(a) RSA approach

The DSA Approach:

- The DSA 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 (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G). The result is a signature consisting of two components, labeled *s* and *r*.
- At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function.
- The verification function also depends on the global public key as well as the sender's public key (PU_a), 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.



(b) DSA approach

The Digital Signature Algorithm

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$ i.e., bit length of N bits

g $= h(p-1)/q \bmod p$, where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature $= (r, s)$

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M = message to be signed

H(M) = hash of M using SHA-1

M', r', s' = received versions of M, r, s

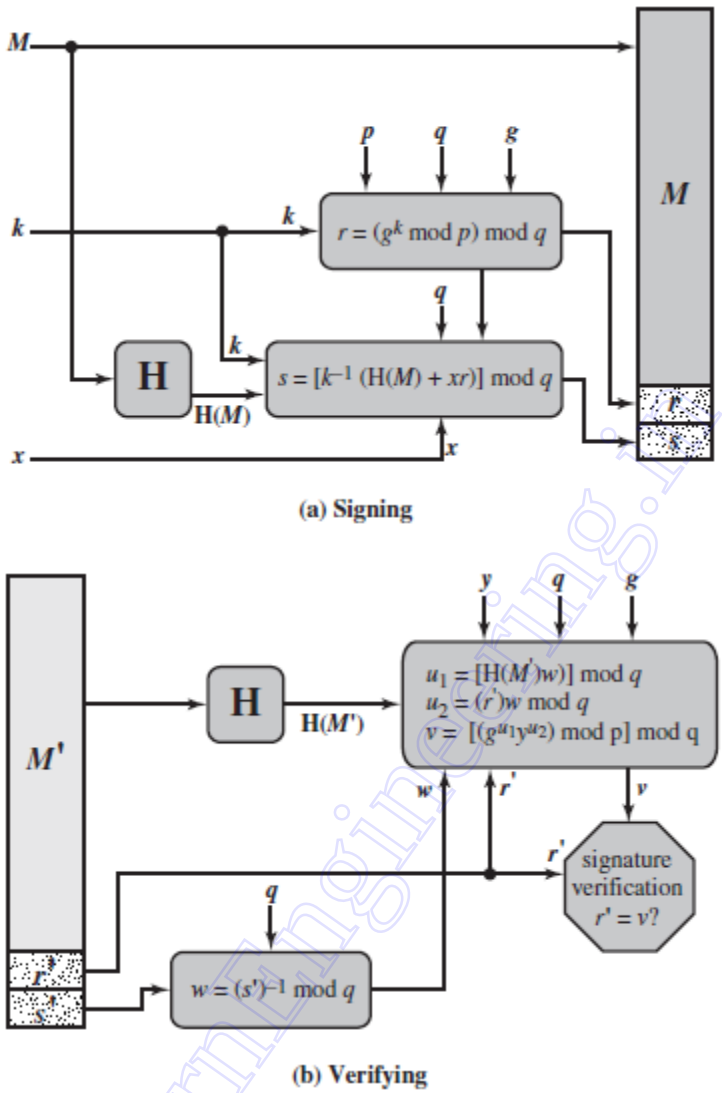


Figure: DSA Signing and Verifying

7. ELGAMAL AND SCHNORR DIGITAL SIGNATURE SCHEMES

- **Discuss about Elgamal and Schnorr digital signature schemes.**
- **Explain digital signaturing with ElGamal public key cryptosystem. (Nov/Dec'13)**

a. ELGAMAL DIGITAL SIGNATURE SCHEME

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.

For a prime number q, if α is a primitive root of q, then

$$\alpha, \alpha^2, \ldots, \alpha^{q-1} \text{ are distinct (mod } q).$$

It can be shown that, if a is a primitive root of α, 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 X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

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

1. Choose a random integer K such that $1 \le K \le q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
2. Compute $S_1 = \alpha^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S_2 = K^{-1} (m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

Any user B can **verify the signature** as follows.

1. Compute $V_1 = \alpha^m \bmod q$.
2. Compute $V_2 = (Y_A)^{S_1} (S_1)^{S_2} \bmod q$.

The signature is valid if $V_1 = V_2$. Assume that the equality is true. Then we have

$\alpha^m \bmod q = (Y_A)^{S_1} (S_1)^{S_2} \bmod q$	assume $V1 = V2$
$\alpha^m \bmod q = \alpha^{X_A S_1} \alpha^{K S_2} \bmod q$	substituting for Y_A and S_1
$\alpha^{m - X_A S_1} \bmod q = \alpha^{K S_2} \bmod q$	rearranging terms
$m - X_A S_1 \equiv K S_2 \bmod (q - 1)$	property of primitive roots
$m - X_A S_1 \equiv K K^{-1} (m - X_A S_1) \bmod (q - 1)$	substituting for $S2$

Example

Assume, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}. We choose $\alpha = 10$.

Alice **generates a key pair** as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^{16} \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is $\{q, a, Y_A\} = \{19, 10, 4\}$.

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. $S_1 = \alpha^K \bmod q = 10^5 \bmod 19 = 3$.
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.

b. SCHNORR DIGITAL SIGNATURE SCHEME

- The Schnorr signature scheme is based on discrete logarithms. The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.
- The main work for signature generation does not depend on the message and can be done during the idle time of the processor.
- The message-dependent part of the signature generation requires multiplying a $2n$-bit integer with an n-bit integer.
- The scheme is based on using a prime modulus p, with $p - 1$ having a prime factor q of appropriate size; that is, $p - 1 \equiv (\bmod q)$. Typically, we use $p \approx 2^{1024}$ and $q \approx 2^{160}$. Thus, p is a 1024-bit number, and q is a 160-bit number, which is also the length of the SHA-1 hash value.

The first part of this scheme is the **generation of a private/public key pair**, which consists of the following steps.

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $\alpha^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

A user with private key s and public key v **generates a signature** as follows.

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$$\boldsymbol{e = H(M \,||\, x)}$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Any other user can **verify the signature** as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = H(M \,||\, x')$.

$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$$

Hence, $\mathbf{H(M \,||\, x') = H(M \,||\, x)}$.

UNIT IV SECURITY PRACTICE & SYSTEM SECURITY 8

Authentication applications – Kerberos – X.509 Authentication services - Internet Firewalls for Trusted System: Roles of Firewalls – Firewall related terminology- Types of Firewalls - Firewall designs – SET for E-Commerce Transactions. Intruder – Intrusion detection system – Virus and related threats – Countermeasures – Firewalls design principles – Trusted systems – Practical implementation of cryptography and security.

PART-A

1. **Define Intruder and List Classes of Intruders. (APR/MAY 2011) (APR/MAY 2010)**

   An individual who gains, or attempts to gain, unauthorized access to a computer system or to gain unauthorized privileges on that system.

   - ✓ Masquerader
   - ✓ Misfeasor
   - ✓ Clandestine user

2. **Write short notes on Intrusion detection system and Malicious software. (NOV/DEC 2011) (APRIL/MAY 2015)**

   A set of automated tools designed to detect unauthorized access to a host system.

   Malicious software:

   Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.

3. **What do you meant by Trojan horse? (APR/MAY 2011)**

   Trojan Horses

   - ✓ A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.
   - ✓ Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly.

4. **Define Firewall. (APRIL/MAY 2015)**

   A firewall is a device or set of devices designed to permit or deny network transmissions based upon a set of rules and is frequently used to protect networks from unauthorized access.

5. **What is worm? (NOV/DEC 2013) (APR/MAY 2015)**

A worm is a self-replicating virus that does not alter files but resides in active memory and duplicates itself. Worms use parts of an operating system that are automatic and usually invisible to the user.

6. List out the requirements of Kerberos. (APR/MAY 2011) (APR/MAY 2010)

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
- **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.
- **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

7. What are the two types of audit records? (MAY/JUNE 2012)

- Native audit records
- Detection specific audit records

8. Define Honeypot . (APR/MAY 2010)

A decoy system designed to lure a potential attacker away from critical systems. It is a form of intrusion detection.

9. What is meant by polymorphic viruses? (APR/MAY 2008)

A virus that mutates with every infection, making detection by the "signature" of the virus impossible.

10. What are zombies? (May/June 2014)

Zombies are programs which secretly takes over another networked computer. Then uses it to indirectly launch attacks. It is often used to launch distributed denial of service (DDoS) attacks. It exploits known flaws in network systems.

PART-B

1. KERBEROS

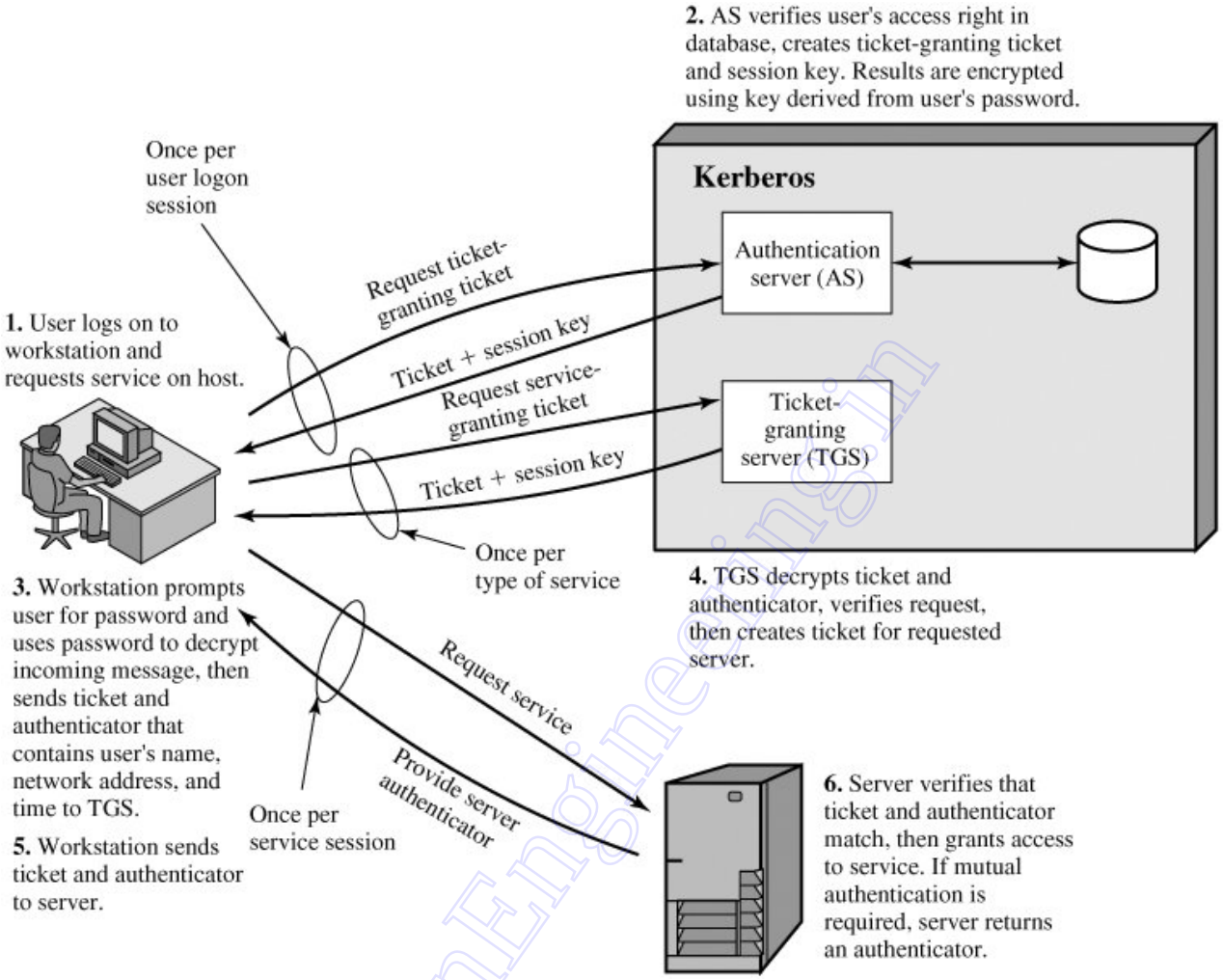
❖ **Explain in detail about Kerberos version 4 and version 5. (May/June 2011)**

Kerberos is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.

Kerberos listed the following requirements:

- ✓ **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
- ✓ **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.
- ✓ **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- ✓ **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Overview of Kerberos



Kerberos Version 4

- ✓ Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service.
- ✓ An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database.

(1) C → AS: $ID_C||P_C||ID_V$

(2) AS →C: **Ticket**

(3) C → V: $ID_C||Ticket$

$$Ticket = E(K_v, [ID_C||AD_C||ID_V])$$

Where
C = client
AS= authentication server
V=server
ID_C= identifier of user on C

ID_V= identifier of V
P_C = password of user on C
AD_C= network address of C
K_v= secret encryption key shared by AS and V

Kerberos Realms and Multiple Kerberi

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a **Kerberos realm**.

Request for Service in Another Realm

(1) C → AS: $ID_c||ID_{tgs}||TS_1$

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

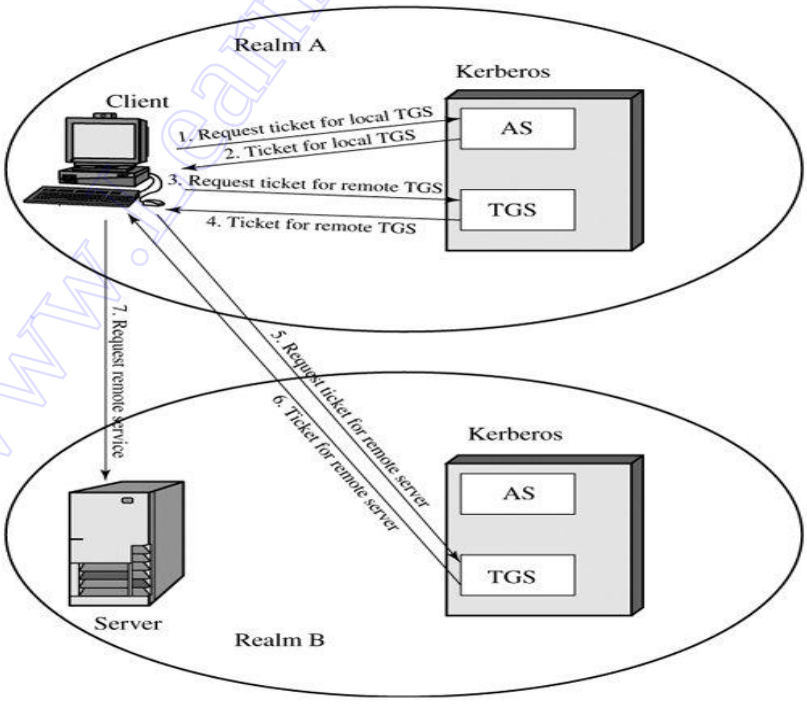
(3) C →TGS: $ID_{tgsrem}||Ticket_{tgs}||Authenticator_c$

(4) TGS → C: $E(K_{c,tgs}, [K_{c,tgsrem}||ID_{tgsrem}||TS_4||Ticket_{tgsrem}])$

(5) C →TGS_{rem}: $ID_{vrem}||Ticket_{tgsrem}||Authenticator_c$

(6) TGS_{rem}→ C: $E(K_{c,tgsrem}, [K_{c,vrem}||ID_{vrem}||TS_6||Ticket_{vrem}])$

(7) C →V_{rem}: $Ticket_{vrem}||Authenticator_c$



Differences between Versions 4 and 5

1. **Encryption system dependence:** Version 4 requires the use of DES. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used.
2. **Internet protocol dependence**: Version 4 requires the use of Internet Protocol (IP) addresses. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.
3. **Message byte ordering:** In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address.
4. **Ticket lifetime:** Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is 2^8 x 5 = 1280 minutes, or a little over 21 hours. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
5. **Authentication forwarding**: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. Version 5 provides this capability.
6. **Interrealm authentication**: In version 4, interoperability among N realms requires on the order of N^2 Kerberos-to-Kerberos relationships. Version 5 supports a method that requires fewer relationships.

2. INTRUSION DETECTION

❖ **Write short notes on Intruders and Explain elaborately on intrusion detection. (May/June 2011, Nov/Dec 2011)**

Three classes of intruders:

- ✓ **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
- ✓ **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
- ✓ **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection
    - ✓ Attempts to copy the password file
    - ✓ Suspicious remote procedure call
    - ✓ Attempts to connect

Intrusion Techniques

- ✓ **One-way function**: The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value.
- ✓ **Access control**: Access to the password file is limited to one or a very few accounts.

The following techniques for learning passwords:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords.
3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
5. Try users' phone numbers, Social Security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.
7. Use a Trojan horse to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

Intrusion Detection

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

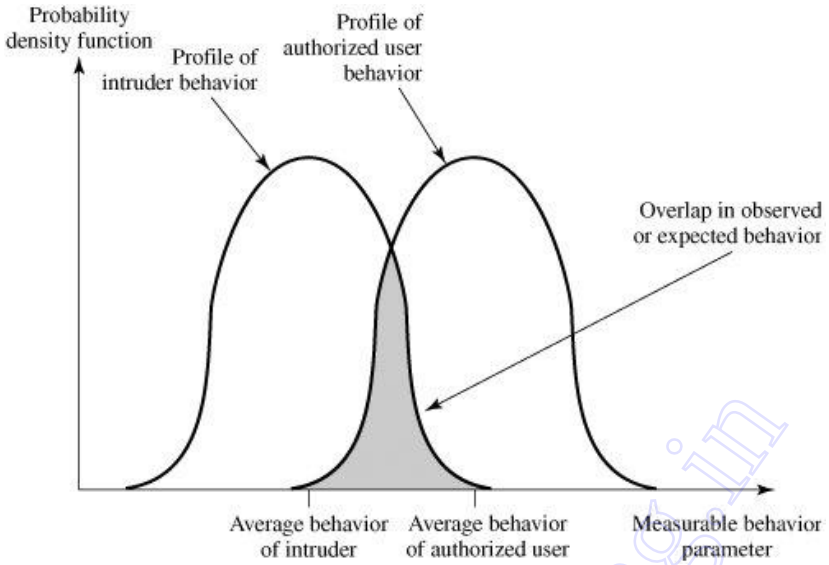


Figure: Profiles of Behavior of Intruders and Authorized Users

Approaches to intrusion detection:

1. **Statistical anomaly detection:** Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.
    a. **Threshold detection:** This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
    b. **Profile based:** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
2. **Rule-based detection:** Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
    a. **Anomaly detection:** Rules are developed to detect deviation from previous usage patterns.
    b. **Penetration identification:** An expert system approach that searches for suspicious behavior.

Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

- **Native audit records:** Virtually all multiuser operating systems include accounting software that collects information on user activity.
- Detection-specific audit records: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system.

Each audit record contains the following fields:

- **Subject**: Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users.
- **Action**: Operation performed by the subject on or with an object; for example, login, read, perform I/O, execute.
- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures.
- **Exception-Condition:** Denotes which, if any, exception condition is raised on return.
- **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource.
- **Time-Stamp:** Unique time-and-date stamp identifying when the action took place.

Statistical Anomaly Detection

Examples of metrics that are useful for profile-based intrusion detection are the following:

- ✓ **Counter:** A nonnegative integer may be incremented but not decremented until it is reset by management action
- ✓ **Gauge:** A nonnegative integer that may be incremented or decremented.
- ✓ **Interval timer:** The length of time between two related events.
- ✓ **Resource utilization:** Quantity of resources consumed during a specified period.

Lists the following approaches that may be taken:

- Mean and standard deviation
- Multivariate
- Markov process
- Time series
- Operational

Rule-Based Intrusion Detection

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

Distributed Intrusion Detection

- ✓ A distributed intrusion detection system may need to deal with different audit record formats.

- ✓ One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network.
- ✓ Either a centralized or decentralized architecture can be used.

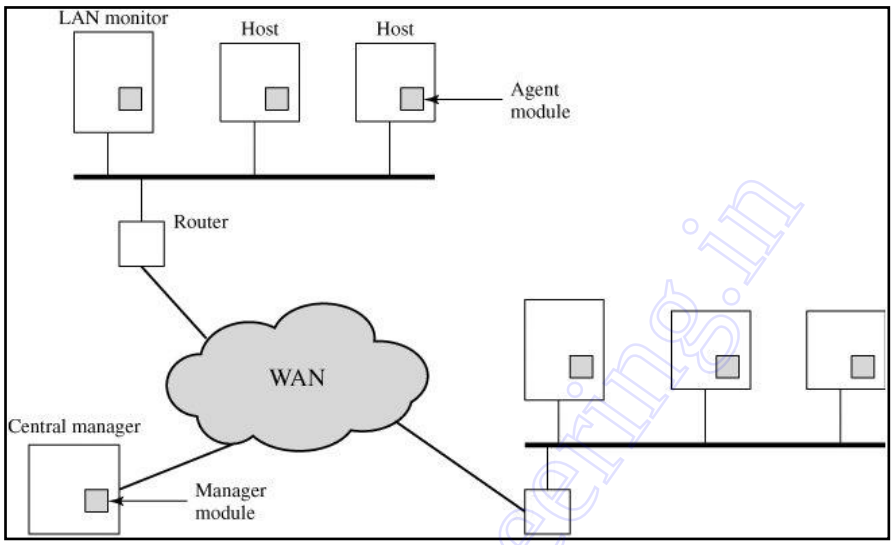


Figure: Architecture for Distributed Intrusion Detection

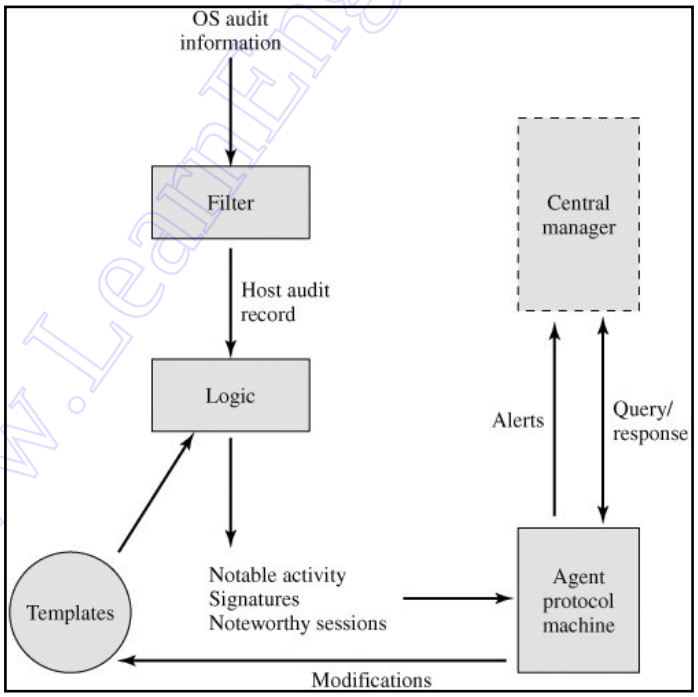


Figure : Agent Architecture

Honeypots

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

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

Password Management

Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password.

- ✓ The ID determines whether the user is authorized to gain access to a system.
- ✓ The ID determines the privileges accorded to the user.
- ✓ The ID is used in what is referred to as discretionary access control.

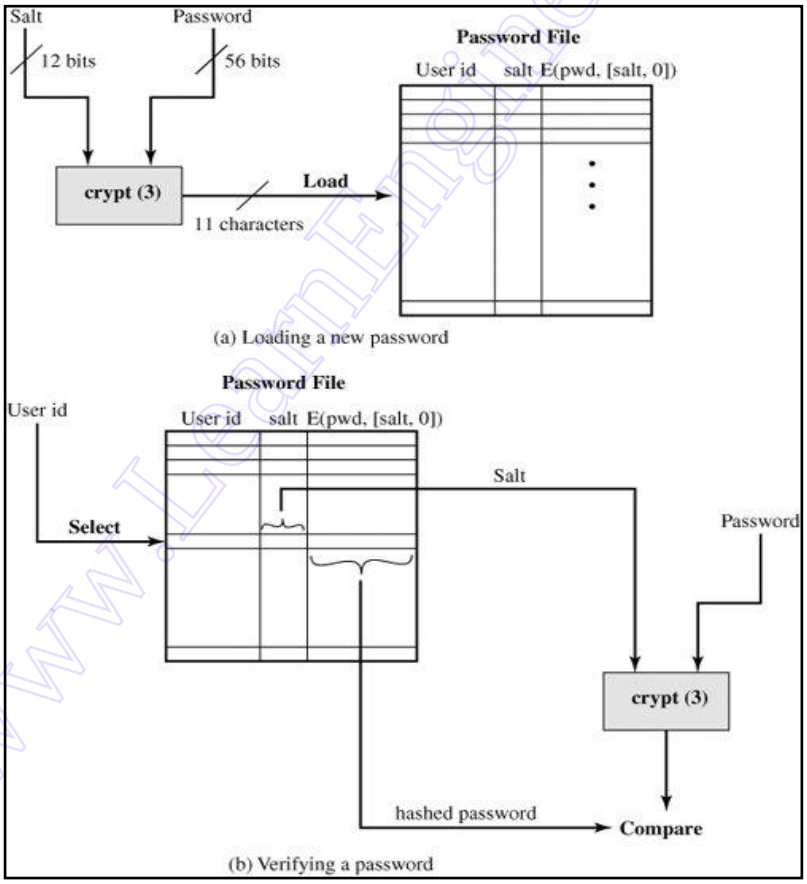


Figure: UNIX Password Scheme

The salt serves three purposes:

- It prevents duplicate passwords from being visible in the password file.
- It effectively increases the length of the password without requiring the user to remember two additional characters.
- It prevents the use of a hardware implementation of DES

Access Control

- Many systems, including most UNIX systems, are susceptible to unanticipated break-ins.
- An accident of protection might render the password file readable, thus compromising all the accounts.
- Some of the users have accounts on other machines in other protection domains, and they use the same password.

Password Selection Strategies

Four basic techniques are in use:

- User education
- Computer-generated passwords
- Reactive password checking
- Proactive password checking

The first approach is a simple system for rule enforcement. For example, the following rules could be enforced:

- All passwords must be at least eight characters long.
- In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks.
- Space: The dictionary must be very large to be effective.
- Time: The time required to search a large dictionary may itself be large.

Two techniques for developing an effective and efficient proactive password checker that is based on rejecting words on a list show promise. One of these develops a Markov model for the generation of guessable passwords

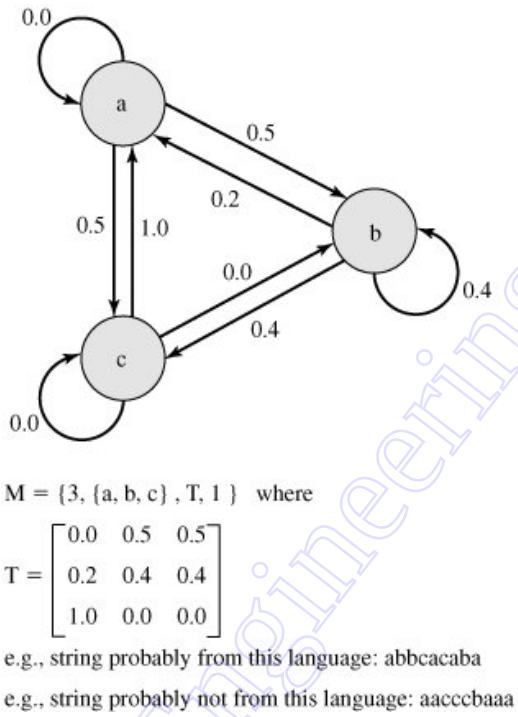


Figure: An Example Markov Model

3. VIRUSES

- **Explain definition, phases, types of virus structures and types of viruses. (April/May 2011, May/June 2011)**

Malicious Programs

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

Terminology of Malicious Programs	
Name	**Description**
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

Backdoor

- ✓ A backdoor, also known as a trapdoor, is a secret entry point into a program that allows someone that is aware of the backdoor to gain access without going through the usual security access procedures.
- ✓ Programmers have used backdoors legitimately for many years to debug and test programs.
- ✓ This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application.

Logic Bomb

- ✓ One of the oldest types of program threat, predating viruses and worms, is the logic bomb.
- ✓ The logic bomb is code embedded in some legitimate program that is set to "explode" when certain conditions are met.
- ✓ Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application. Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.

Trojan Horses

- A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.
- Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly.

- For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changed the invoking user's file permissions so that the files are readable by any user.

Zombie

- ✓ A zombie is a program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the zombie's creator.
- ✓ Zombies are used in denial-of-service attacks, typically against targeted Web sites.
- ✓ The zombie is planted on hundreds of computers belonging to unsuspecting third parties, and then used to overwhelm the target Web site by launching an overwhelming onslaught of Internet traffic.

The Nature of Viruses

- ✓ A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

During its lifetime, a typical virus goes through the following four phases:

- **Dormant phase**: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
- **Propagation phase**: The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
- **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.
- **Execution phase:** The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Virus Structure

- ✓ A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.
- ✓ The virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program.

```
     program V :=

{goto main;
     1234567;

     subroutine infect-executable :=
          {loop:
          file := get-random-executable-file;
          if (first-line-of-file = 1234567)
             then goto loop
             else prepend V to file; }

     subroutine do-damage :=
          {whatever damage is to be done}

     subroutine trigger-pulled :=
          {return true if some condition holds}

main:    main-program :=
          {infect-executable;
          if trigger-pulled then do-damage;
          goto next;}
next:

}
```

Figure: A Simple Virus

```
program CV :=

{goto main;
    01234567;

    subroutine infect-executable :=
            {loop:
                file := get-random-executable-file;
            if (first-line-of-file = 01234567) then goto loop;
        (1)  compress file;
        (2)  prepend CV to file;
        }

main:   main-program :=
            {if ask-permission then infect-executable;
        (3)  uncompress rest-of-file;
        (4)  run uncompressed file;}
        }
```

Figure: A Compression Virus

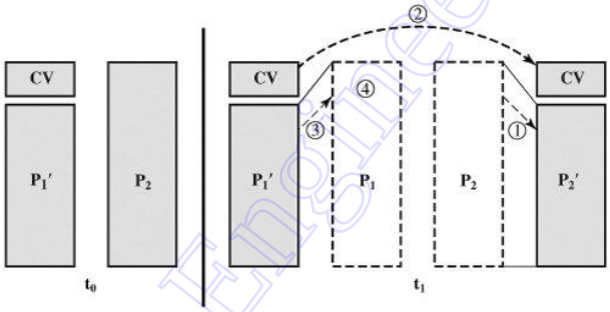


Figure: Logic for a Compression Virus

Initial Infection

✓ Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes.

Types of Viruses

- Parasitic virus: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.

- Memory-resident virus: Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.
- Boot sector virus: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
- Stealth virus: A form of virus explicitly designed to hide itself from detection by antivirus software.
- Polymorphic virus: A virus that mutates with every infection, making detection by the "signature" of the virus impossible.
- Metamorphic virus: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.

Macro Viruses

1. A macro virus is platform independent.
2. Macro viruses infect documents, not executable portions of code.
3. Macro viruses are easily spread. A very common method is by electronic mail.

E-mail Viruses

1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
2. The virus does local damage.

Worms

- ✓ A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again.
- ✓ An e-mail virus has some of the characteristics of a worm, because it propagates itself from system to system. However, we can still classify it as a virus because it requires a human to move it forward.
- ✓ A worm actively seeks out more machines to infect and each machine that is infected serves as an automated launching pad for attacks on other machines.

Examples include the following:

- **Electronic mail facility**: A worm mails a copy of itself to other systems.
- **Remote execution capability**: A worm executes a copy of itself on another system.
- **Remote login capability**: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

1. Search for other systems to infect by examining host tables or similar repositories of remote system addresses.
2. Establish a connection with a remote system.
3. Copy itself to the remote system and cause the copy to be run.

The Morris Worm

For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. To obtain the passwords, the worm ran a password-cracking program that tried
    a. Each user's account name and simple permutations of it
    b. A list of 432 built-in passwords that Morris thought to be likely candidates
    c. All the words in the local system directory
2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.
3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

Recent Worm Attacks

Nimda spreads by multiple mechanisms:

- from client to client via e-mail
- from client to client via open network shares
- from Web server to client via browsing of compromised Web sites

- from client to Web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities
- from client to Web server via scanning for the back doors left behind by the "Code Red II" worms

State of Worm Technology

The state of the art in worm technology includes the following:

- **Multiplatform**: Newer worms are not limited to Windows machines but can attack a variety of platforms, especially the popular varieties of UNIX.
- **Multiexploit:** New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications.
- **Ultrafast spreading**: One technique to accelerate the spread of a worm is to conduct a prior Internet scan to accumulate Internet addresses of vulnerable machines.
- **Polymorphic**: To evade detection, skip past filters, and foil real-time analysis, worms adopt the virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.
- **Metamorphic:** In addition to changing their appearance, metamorphic worms have a repertoire of behavior patterns that are unleashed at different stages of propagation.
- **Transport vehicles**: Because worms can rapidly compromise a large number of systems, they are ideal for spreading other distributed attack tools, such as distributed denial of service zombies.
- **Zero-day exploit**: To achieve maximum surprise and distribution, a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched.

Virus Countermeasures

Antivirus Approaches

- **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.
- **Identification:** Once detection has been achieved, identify the specific virus that has infected a program.
- **Removal:** Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.

Four generations of antivirus software:

- **First generation**: simple scanners
- **Second generation**: heuristic scanners
- **Third generation**: activity traps
- **Fourth generation**: full-featured protection

Advanced Antivirus Techniques

Generic Decryption

✓ Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds.

✓ In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

- **CPU emulator**: A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor.
- **Virus signature scanner**: A module that scans the target code looking for known virus signatures.
- **Emulation control module**: Controls the execution of the target code.

Digital Immune System

- Integrated mail systems: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.
- Mobile-program systems: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

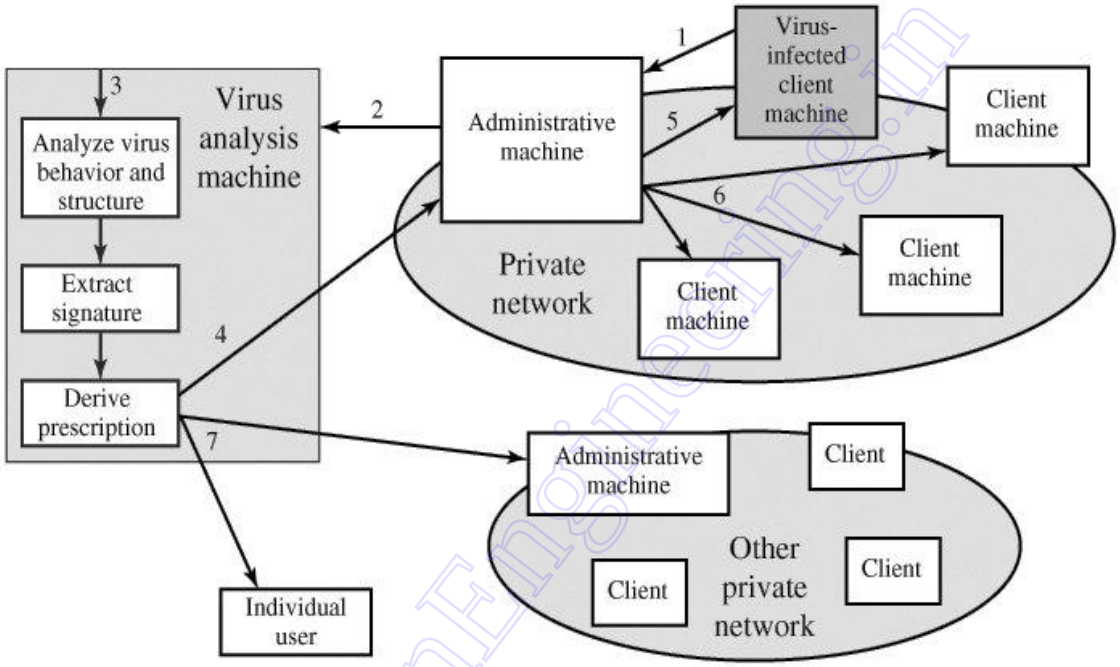


Figure: Digital Immune System

4. FIREWALL

❖ **Explain in detail about Firewall design principles explain types of firewalls in detail. (April/May 2011, May/June 2011, Nov/Dec 2011)**

Firewall Design Principles

- Centralized data processing system, with a central mainframe supporting a number of directly connected terminals
- Local area networks (LANs) interconnecting PCs and terminals to each other and the mainframe
- Premises network, consisting of a number of LANs, interconnecting PCs, servers, and perhaps a mainframe or two
- Enterprise-wide network, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN)
- Internet connectivity, in which the various premises networks all hook into the Internet and may or may not also be connected by a private WAN

Firewall Characteristics

1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible, as explained later in this section.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies, as explained later in this section.
3. The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system.

Firewalls focused primarily on service control, but they have since evolved to provide all four:

- **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound.
- **Direction control**: Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.
- **User control**: Controls access to a service according to which user is attempting to access it.
- **Behavior control**: Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

Capabilities are within the scope of a firewall:

1. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
2. A firewall provides a location for monitoring security-related events.
3. A firewall is a convenient platform for several Internet functions that are not security related.
4. A firewall can serve as the platform for IPSec.

Firewalls have their limitations, including the following:

1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP.
2. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
3. The firewall cannot protect against the transfer of virus-infected programs or files.

Types of Firewalls

Packet-Filtering Router

- ✓ A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet.
- ✓ Filtering rules are based on information contained in a network packet:
  - **Source IP address:** The IP address of the system that originated the IP packet (e.g., 192.178.1.1)
  - **Destination IP address:** The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2)
  - **Source and destination transport-level address:** The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET
  - **IP protocol field:** Defines the transport protocol
  - **Interface:** For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for.

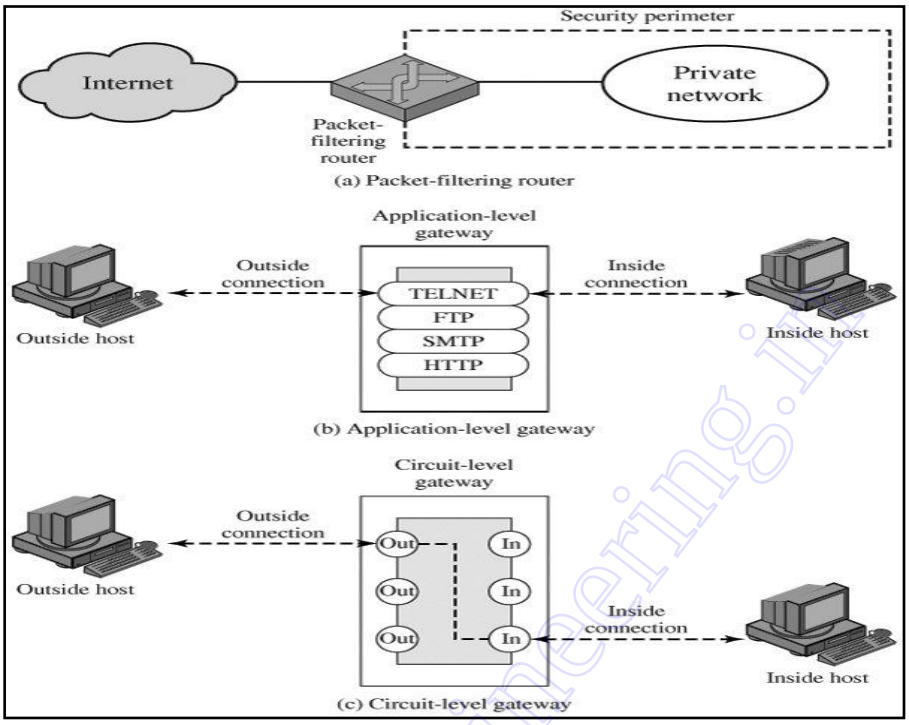


Figure: Firewall Types

Two default policies are possible:

- **Default = discard**: That which is not expressly permitted is prohibited.
- **Default = forward**: That which is not expressly prohibited is permitted.

Some of the attacks that can be made on packet-filtering routers and the appropriate countermeasures are thefollowing:

- **IP address spoofing**: The intruder transmits packets from the outside with a source IP address field containing an address of an internal host.
- **Source routing attacks**: The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. The countermeasure is to discard all packets that use this option.

- **Tiny fragment attacks**: The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment.

Stateful Inspection Firewalls

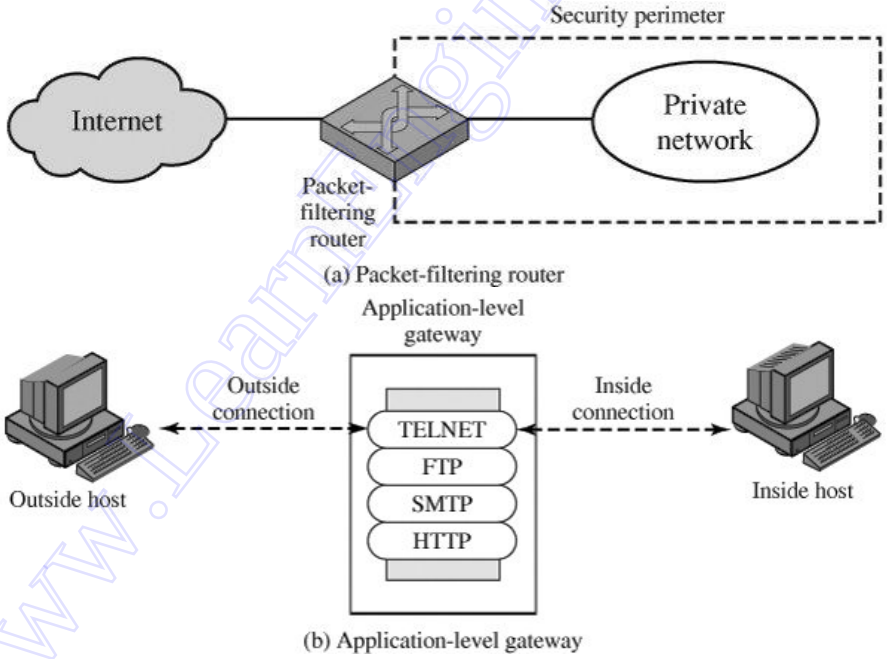
Circuit-Level Gateway

✓ A circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host.

Bastion Host

- The bastion host hardware platform executes a secure version of its operating system, making it a trusted system.

Firewall Configurations



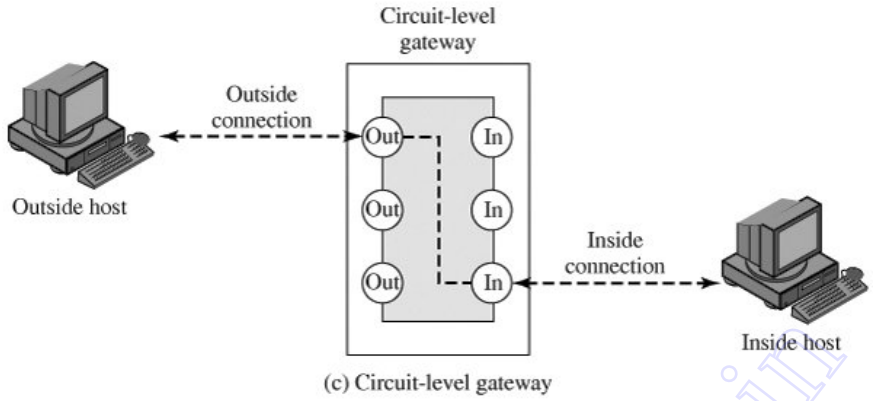


Figure: Firewall Configurations

Typically, the router is configured so that

1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.
2. For traffic from the internal network, only IP packets from the bastion host are allowed out.

5. TRUSTED SYSTEMS

❖ **Explain in detail about Trusted Systems (MAY/JUNE 2012)**

Data Access Control

The basic elements of the model are as follows:

- **Subject**: An entity capable of accessing objects. Generally, the concept of subject equates with that of process.
- **Object:** Anything to which access is controlled. Examples include files, portions of files, programs, and segments of memory.
- **Access right**: The way in which an object is accessed by a subject. Examples are read, write, and execute.

	Program1	...	SegmentA	SegmentB
Process1	Read Execute		Read Write	
Process2				Read
. . .				

(a) Access matrix

Access control list for Program1:
Process1 (Read, Execute)

Access control list for SegmentA:
Process1 (Read, Write)

Access control list for SegmentB:
Process2 (Read)

(b) Access control list

Capability list for Process1:
Program1 (Read, Execute)
SegmentA (Read, Write)

Capability list for Process2:
Segment B (Read)

(c) Capability list

Figure: Access Control Structure

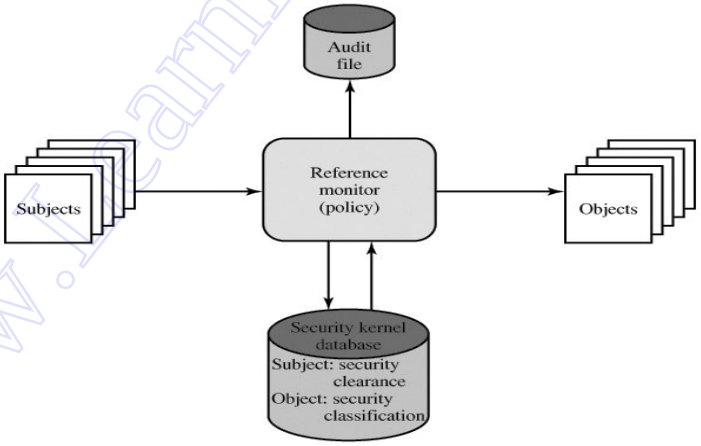


Figure: Reference Monitor Concept

Trojan Horse Defense

- ✓ One way to secure against Trojan horse attacks is the use of a secure, trusted operating system.
- ✓ In this case, a Trojan horse is used to get around the standard security mechanism used by most file management and operating systems: the access control list

UNIT V E-MAIL, IP & WEB SECURITY 9

E-mail Security: Security Services for E-mail-attacks possible through E-mail - establishing keys privacy-authentication of the source-Message Integrity-Non-repudiation-Pretty Good Privacy-S/MIME. IPSecurity: Overview of IPSec - IP and IPv6-Authentication Header-Encapsulation Security Payload (ESP)-Internet Key Exchange (Phases of IKE, ISAKMP/IKE Encoding). Web Security: SSL/TLS Basic Protocol-computing the keys- client authentication-PKI as deployed by SSLAttacks fixed in v3-Exportability-Encoding-Secure Electronic Transaction (SET).

PART-A

1. **What is tunnel mode in IPsecurity? Apr/May'15**

   Tunnel Mode

   1. It provide the protection for entire IP Packet
   2. ESP in this mode encrypt authenticate the entire IP packet.
   3. AH in this mode authenticate the entire IP Packet plus selected portion of outer IP Header.

2. **List out the services provided by PGP. May/June'13**
   - Digital signature
   - Message encryption
   - Compression
   - E-mail compatibility

3. **Expand and define SPI. May/June'13**

   Security Parameter Index (SPI)

   A 32-bit unsigned integer assigned to this SA and having local significance only. The SPI is carried in **Authentication Header (AH)** and **Encapsulating Security Payload(**ESP) headers to enable the receiving system to select the security association (SA) under which a received packet will be processed.

4. Define: SET. Nov/Dec'2013.

Secure Electronic Transaction (SET) is an open encryption and security specification designed to protect credit card transactions on the Internet.

- ✓ Confidentiality of information
- ✓ Integrity of data
- ✓ Cardholder account authentication
- ✓ Merchant authentication

5. What are the different types of MIME? Nov/Dec'12

- Text
  - Plain and Enriched
- Multipart
  - Mixed
  - Parallel
  - Alternative
  - Digest
- Message
  - rfc822.
  - Partial
  - External-body.
- Image (jpeg and gif)
- Video mpeg MPEG format.
- Audio (Basic)
- Application (PostScript Adobe Postscript format).

6. What protocol comprises SSL? Nov/Dec'12

SSL is a general-purpose service implemented as a set of protocols that rely on TCP.

7. Why does ESP include a padding field? May/June'2012

Padding- Variable length, 0 to 255bytes.Padding may be required, irrespective of encryption algorithm requirements, to ensure that the resulting cipher text terminates on a 4 byte boundary. Specifically, the Pad length and Next header fields must be right aligned within a 4 byte word to ensure that the Authentication data field, if present, is aligned on a 4 byte boundary. Pad length. 8 bits. Specifies the size of the Padding field in bytes.

8. Define TLS.(IT2352 / May/June'12)

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL. TLS is defined as a Proposed Internet Standard in RFC 5246. RFC 5246 is very similar to SSLv3.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 number.

9. What do you mean by S/MIME? (IT2352 / May/June'12)

Secure/Multipurpose Internet Mail Extension is an Internet standard approach to e-mail security that incorporates the same functionality as PGP.Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. PGP and S/MIME are on an IETF standards track.

10. Mention four SSL protocols. (Apr/May'11)

- SSL Handshake Protocol
- SSL Change Cipher Spec Protocol
- SSL Alert Protocol
- HTTP
- SSL Record Protocol
- TCP and IP

11. Mention the fields of IPSec authentication header. (Apr/May'10)

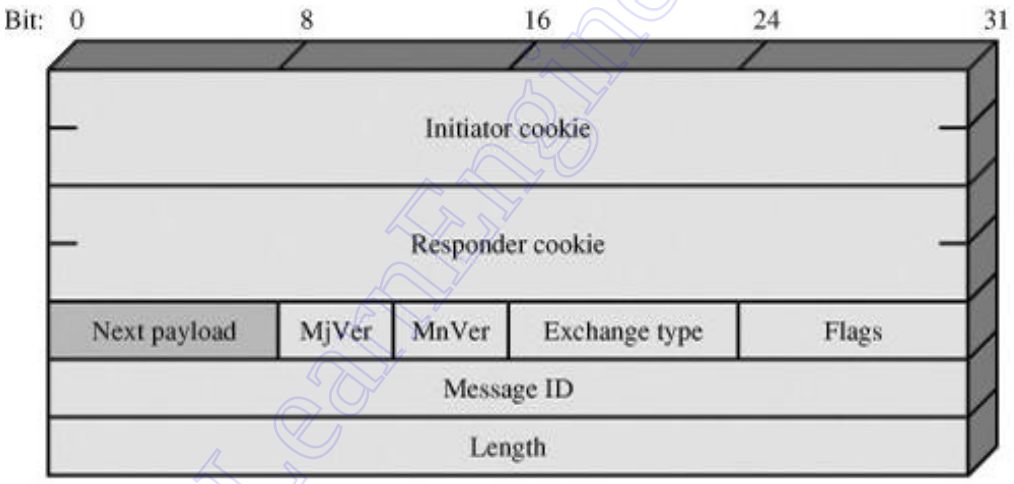
- Next Header Length

- Security Parameters Index (SPI)
- Sequence Number
- Authentication Data

12. Why the leading two octets of digest are stored in PGP messages along with encrypted message digest? (R-2004) (Apr/May'08)

Leading two octets of message digest: L,to enable the recipient to determine if the correct public key (K(A)e) was used to decrypt the message digest for authentication, by comparing this plain text 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, for the authentication and error detection.

13. Draw the header format for an ISAKMP message. May/June'2007



(a) ISAKMP header

PART-B

1. PREETY GOOD PRIVACY

- **Explain Pretty Good Privacy in detail. (16 Marks) May/June'14,Nov/Dec'12**
- **Explain PGP message generation and reception.(16 Marks) Apr/May'11**
- **Illustrate the confidentiality service provided by PGP.(8 Marks) (May/June'2007)**
- **For what purpose Zimmerman developed PGP? Brief the various services provided by PGP. Discuss the threats faced by an e-mail and explain its security requirements to provide a secure e-mail service. (16 Marks) (Nov/Dec '14)**

Pretty Good Privacy

Definition of PGP: **(2 Marks Nov/Dec'2013)**

PGP provides confidentiality and authentication service that can be used for electronic mail and file storage applications**.**

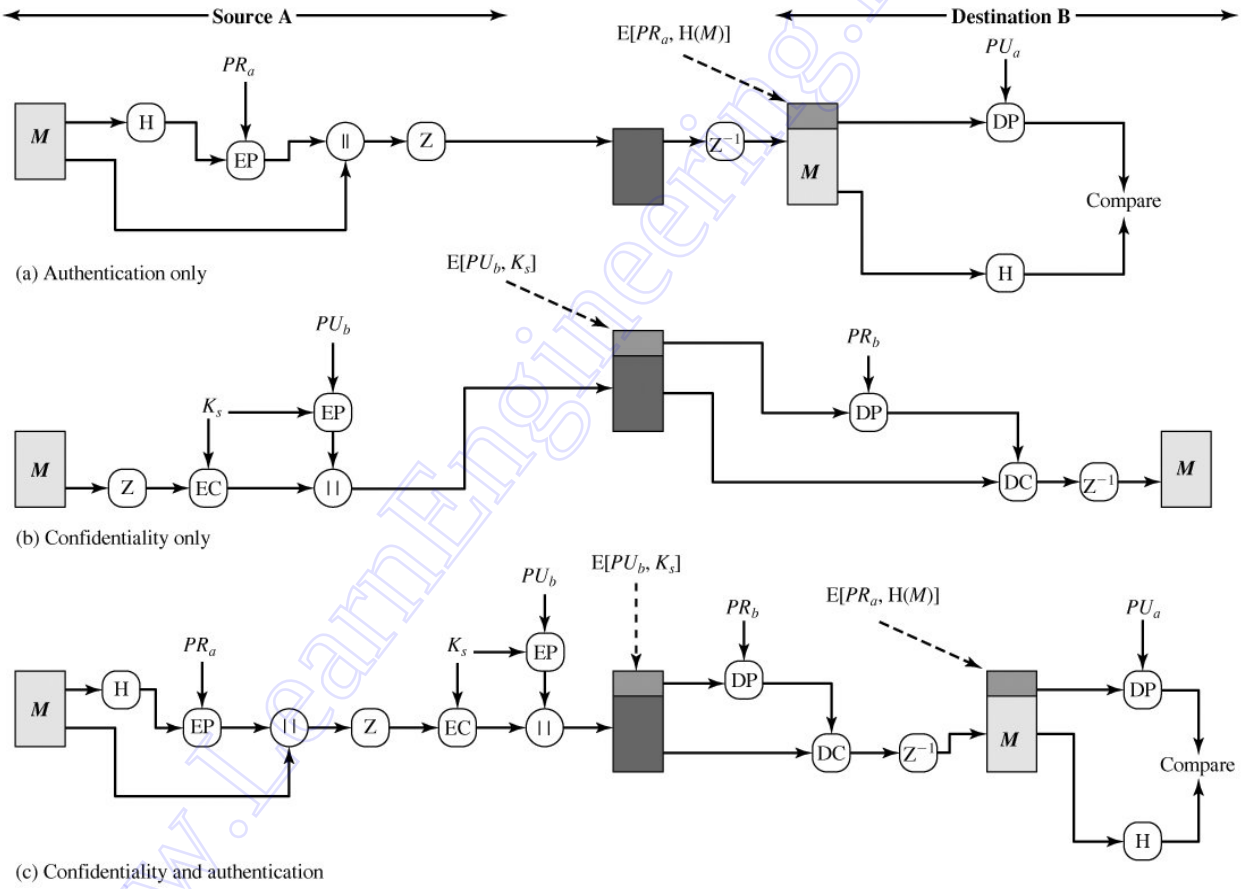
Pretty Good Privacy is an open-source freely available software package for e-mail security. It provides authentication through the use of digital signature; confidentiality through the use of symmetric block encryption; compression using the ZIP algorithm; e-mail compatibility using the radix-64 encoding scheme; and segmentation and reassembly to accommodate long e-mails.

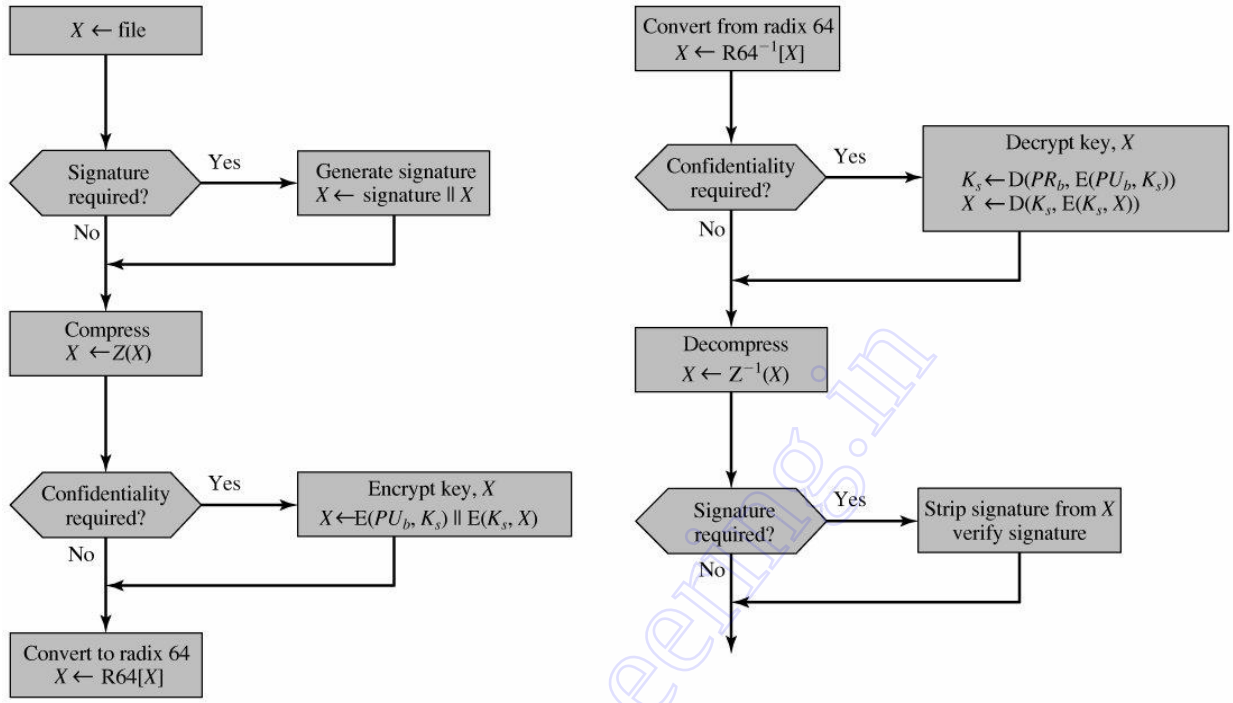
PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

1. Selected the best available cryptographic algorithms as building blocks
2. Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to- use commands

3. Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as AOL (America On Line)
4. Entered into an agreement with a company (Viacrypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP.

Confidentiality and Authentication





(a) Generic transmission diagram (from A)

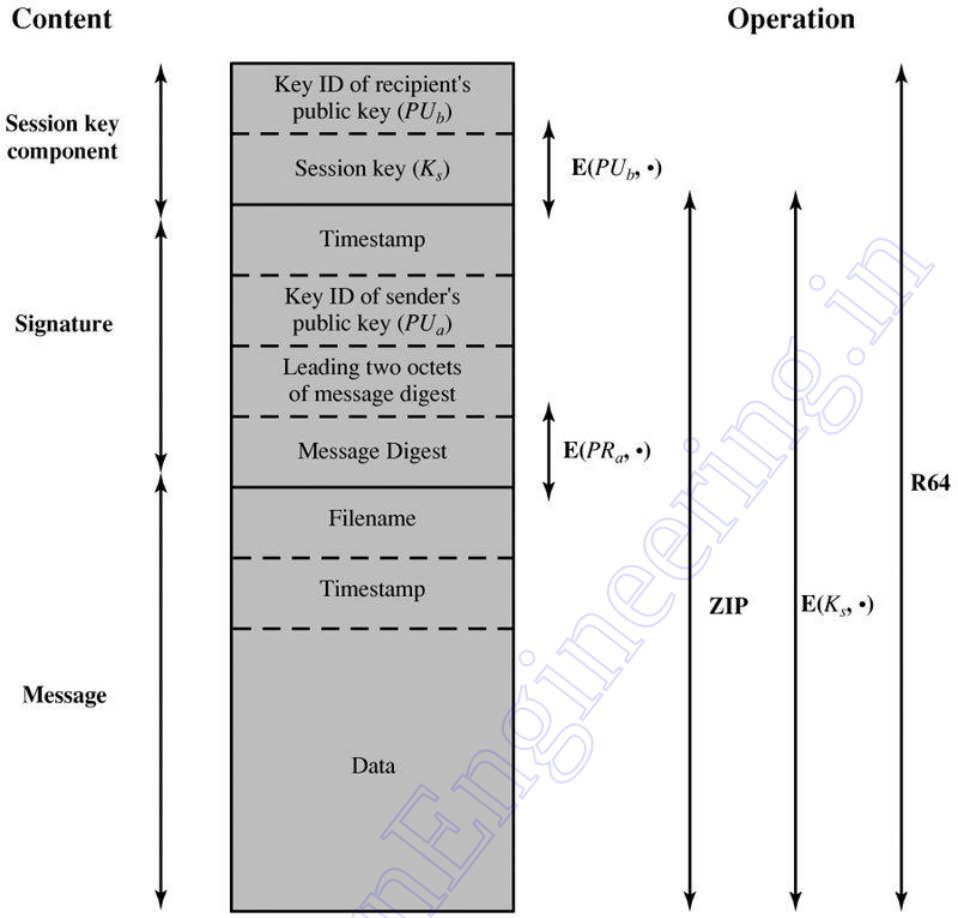
(b) Generic reception diagram (to B)

Cryptographic Keys and Key Rings

1. A means ofType equation here. generating unpredictable session keys is needed.
2. We would like to allow a user to have multiple public-key/private-key pairs.
3. Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

General Format of PGP Message (from A to B)

Sketch the general format for PGP message. (2 Marks-Nov/Dec'2014)



Notation:

E(PU_b, •)	=	encryption with user b's public key
E(PR_a, •)	=	encryption with user a's private key
E(K_s, •)	=	encryption with session key
ZIP	=	Zip compression function
R64	=	Radix-64 conversion function

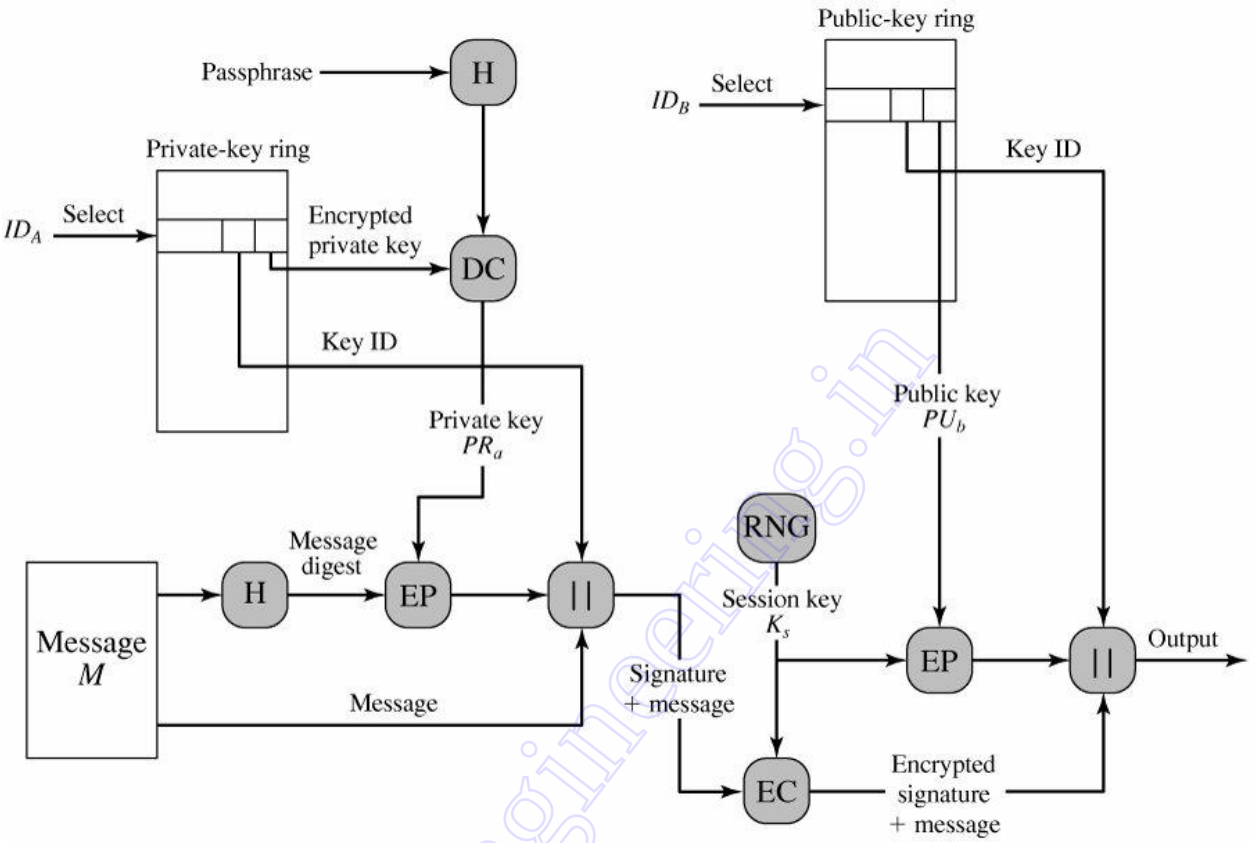


Figure: PGP Message Reception (from User A to User B; no compression or radix 64 conversion)

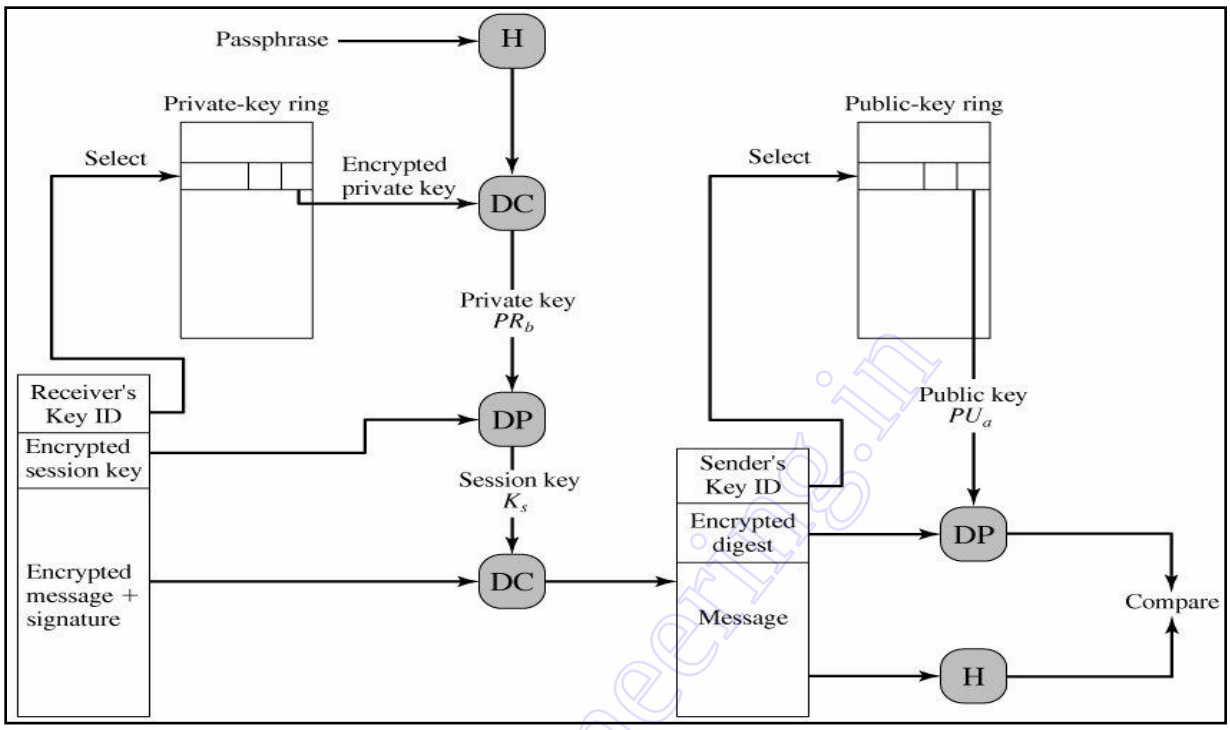
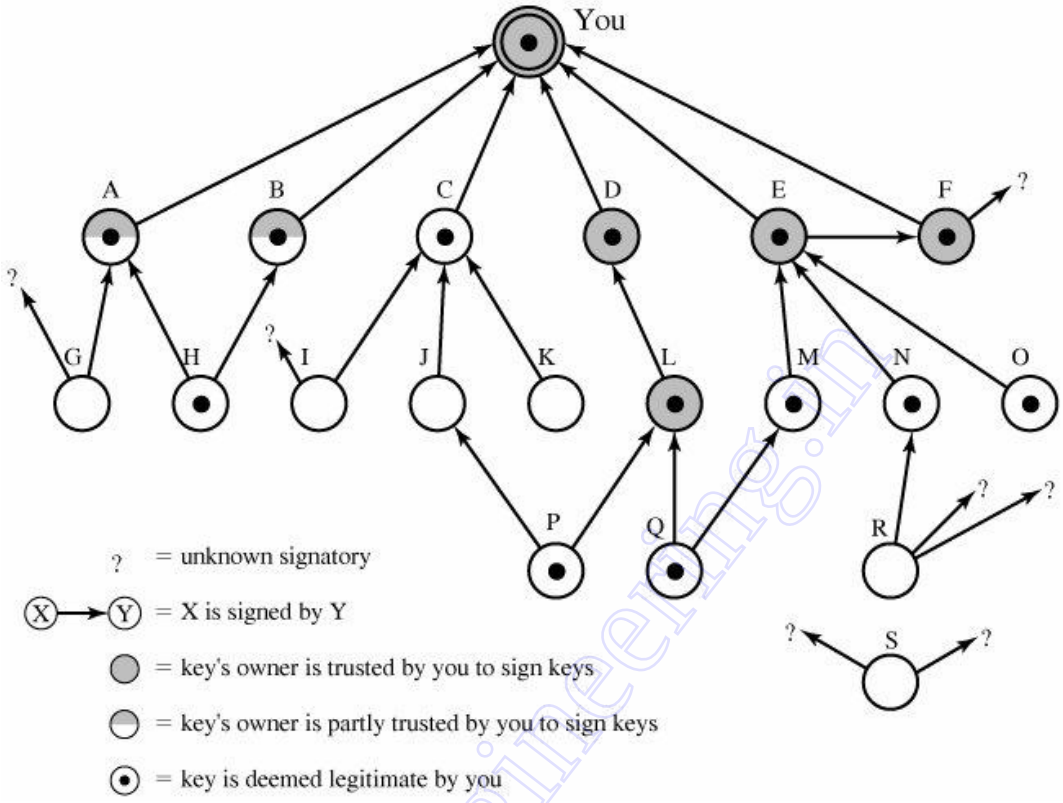


Figure: PGP Trust Model Example



2. WEB SECURITY

- **Write short notes about Nov/Dec'2013**
  - **a. Web security (8 Marks)**
  - **b. SSL (8 Marks)**
- **Write about SSL and TLS. (16 Marks) Apr/May'15, May/June'2012.**
- **Explain about the PKI. (16 Marks) Nov/Dec'2013**

a. WEB SECURITY:

Web Security Considerations:

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. Secure socket layer (SSL) provides security

services between TCP and applications that use TCP. The Internet standard version is called transport layer service (TLS).

The Web presents new challenges not generally appreciated in the context of computer and network security:

- The Internet is two way. The electronic publishing systems involving Teledex, 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 for corporate, product information and for business transactions. Reputations can be damaged and money can be lost if the Web servers are disrupted.
- 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.
- Casual and untrained (in security matters) users are common clients for Web-based services.Such users are not necessarily aware of the security.

Web Security Threats:

Integrity

- Modification of user data
- Trojan horse browser
- Modification of memory
- Modification of message traffic in transit

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

Denial of service

- Killing of user threads

- Flooding machine with bogus requests
- Filling up disk or memory
- Isolating machine by DNS attacks

Authentication

- Impersonation of legitimate users
- Data forgery

b. SSL:

SSL is a layered protocol. It is two layer protocols. At the lower level, the SSL Record Protocol is layered on top of some reliable transport protocol such as TCP.

The Hypertext Transfer Protocol (HTTP), which provides a transfer service for Web client/server interaction, can operate on top of the SSL Record Protocol.

The SSL Record Protocol takes the upper-layer application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies an MAC, encrypts, adds a header, and transmits the result to TCP.

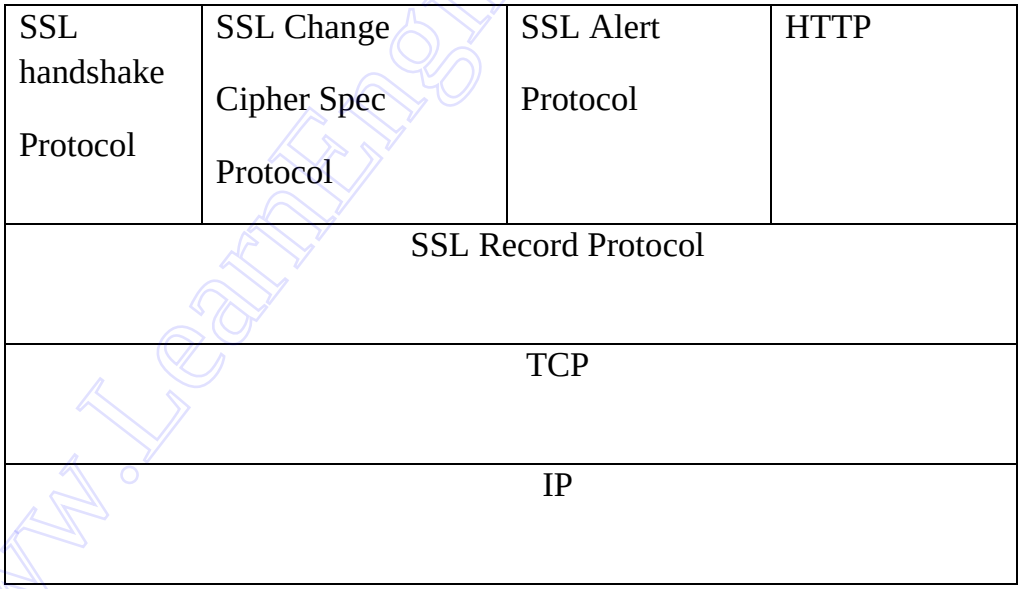


Figure: Two-layered SSL protocols.

Session and Connection States:

There are two defined specifications: SSL session and SSL connection.

SSL session:

An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. They define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.
The session state is defined by the following elements:

- Session identifier
- Peer certificate
- Compression method
- Cipher spec
- Master secret
- Is resumable

SSL connection:

A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
The connection state is defined by the following elements:

- Server and client random
- Server write MAC secret
- Client write MAC secret
- Server write key
- Client write key
- Initialisation vectors
- Sequence numbers

SSL Record Protocol:

The SSL Record Protocol provides basic security services to various higher-layer protocols. Three upper-layer protocols are defined as part of SSL: the Handshake Protocol, the Change Cipher Spec Protocol and the Alert Protocol.

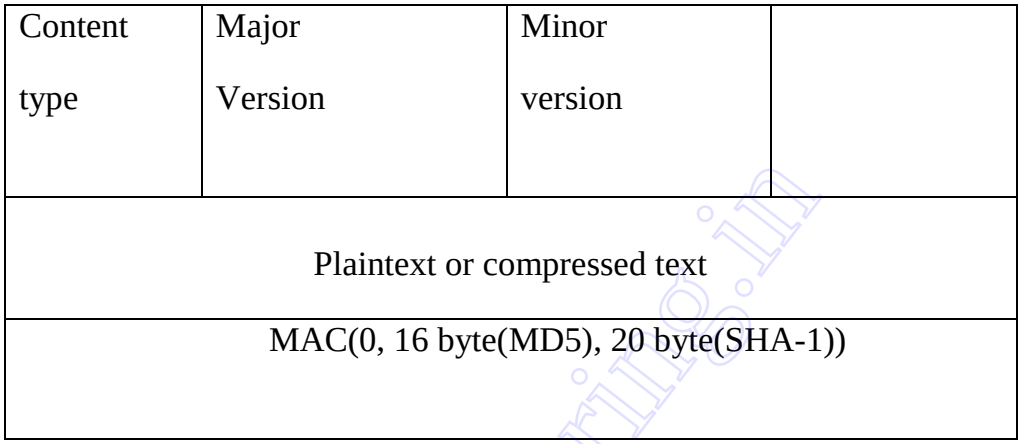


Figure: SSL Record Protocol format.

The composed fields consist of:

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

SSL Alert Protocol:

One of the content types supported by the SSL Record Layer is the alert type. Alert messages convey the severity of the message and a description of the alert. Alert messages consist of 2 bytes. The first byte takes the value warning or fatal to convey the seriousness of the message. If the level is fatal, SSL immediately terminates the connection.

SSL Handshake Protocol:

The SSL Handshake Protocol operated on top of the SSL Record Layer is the most important part of SSL. This protocol provides three services for SSL connections between the server and client. The Handshake Protocol

1. Allows the client/server to agree on a protocol version.
2. To authenticate each other by forming an MAC.
3. To negotiate an encryption algorithm and cryptographic keys for protecting data sent in an SSL record before the application protocol transmits or receives its first byte of data.

The Handshake Protocol consists of a series of messages exchanged by the client and server.

Client server

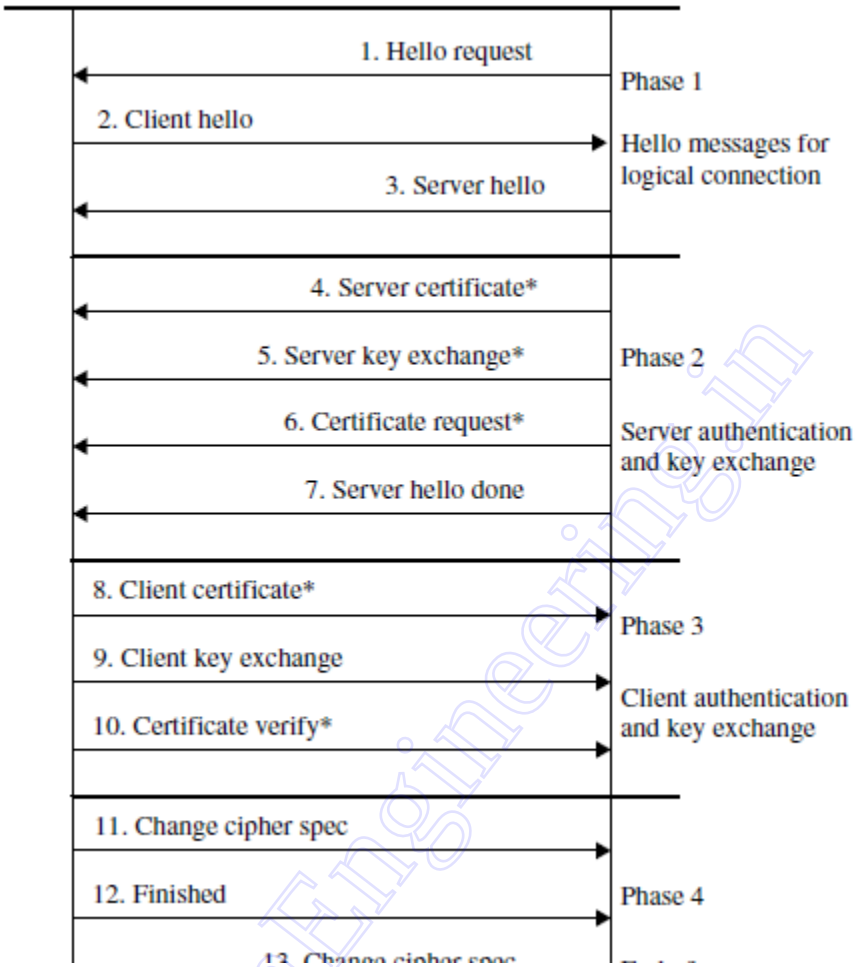


Figure: Handshake Protocol

c. TLS:

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL.Transport Layer Security is defined as a Proposed Internet Standard in RFC 2246. RFC 2246 is very similar to SSLv3. 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 number.

Version Number:

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

Message Authentication Code:

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

HMAC is defined as follows:

$$HMACK(M) = H[(K+ opad)||H[(K+ ipad)||M]]$$

where

H = embedded hash function (for TLS, either MD5 or SHA-1)

M = message input to HMAC

K+ = secret key padded with zeros on the left so that the result is equal to the block length of the hash code(for MD5 and SHA-1, block length = 512 bits)

ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)

opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key.

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

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

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

Pseudorandom Function:

TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The objective is to make use of a relatively small shared secret value but to generate

longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs.

The PRF is based on the following data expansion

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.

To make PRF as secure as possible, it uses two hash algorithms in a way that should guarantee its security if either algorithm remains secure.

PRF is defined as

PRF(secret, label, seed) = P_MD5(S1, label || seed)
P_SHA-1(S2, label || seed)

PRF takes as input a secret value, an identifying label, and a seed value and produces an output of arbitrary length.

Alert Codes:

TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate. Additional codes are defined in TLS

- decryption_failed: A ciphertext decrypted in an invalid way;
- record_overflow: A TLS record was received with a payload (ciphertext) whose length exceeds 214 + 2048 bytes.
- unknown_ca: A valid certificate chain or partial chain was received trusted CA.
- access_denied: A valid certificate was received, but when access control was applied, the sender decided not to proceed with the negotiation.
- decode_error: A message could not be decoded because a field was out of its specified range.
- export_restriction: A negotiation not in compliance with export

restrictions on key length was detected.

- protocol_version: The protocol version the client attempted to negotiate is recognized but not supported.
- insufficient_security: Returned instead of handshake_failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client.
- internal_error: An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue.

New alerts include the following:

- decrypt_error: A handshake cryptographic operation failed, including being unable to verify a signature, decrypt a key exchange, or validate a finished message.
- user_canceled: This handshake is being canceled for some reason unrelated to a protocol failure.
- no_renegotiation: Sent by a client in response to a hello request or by the server in response to a client hello after initial handshaking.

Cipher Suites:

There are several small differences between the cipher suites available under SSLv3 and under TLS:

- Key Exchange: TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.
- Symmetric Encryption Algorithms: TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.

Client Certificate Types:

TLS defines the following certificate types to be requested in a certificate_request message: rsa_sign,dss_sign, rsa_fixed_dh, and dss_fixed_dh. These are all defined in SSLv3. In addition, SSLv3 includes rsa_ephemeral_dh, dss_ephemeral_dh, and fortezza_kea.

Certificate_Verify and Finished Messages:

In the TLS certificate_verify message, the MD5 and SHA-1 hashes are calculated only over handshake_messages.

For TLS, we have

PRF(master_secret, finished_label, MD5(handshake_messages)||

SHA-1(handshake_messages))

where finished_label is the string "client finished" for the client and "server finished" for the server.

Cryptographic Computations:

The pre_master_secret for TLS is calculated in the same way as in SSLv3.The master_secret in TLS is calculated as a hash function of the pre_master_secret and the two hello random numbers. The form of the TLS calculation is different from that of SSLv3 and is defined as follows:

master_secret = PRF(pre_master_secret, "master secret", ClientHello.random || ServerHello.random)

The algorithm is performed until 48 bytes of pseudorandom output are produced. The calculation of the key block material (MAC secret keys, session encryption keys, and IVs) is defined as follows:

key_block = PRF(master_secret, "key expansion",

SecurityParameters.server_random ||

SecurityParameters.client_random)

until enough output has been generated.

Padding:

In SSL, the padding added prior to encryption of user data is the minimum amount required so that the total size of the data to be encrypted is a multiple of the cipher's block length. In TLS, the padding can be any amount that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes.

d. **PKI.**

A public key infrastructure (PKI) is defined as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography. Typically, PKI implementations make use of X.509 certificates so PKIX model.

Principal objective:

- Develop a PKI is to enable secure, convenient, and efficient acquisition of public keys.

Key elements of the PKIX model:

These elements are

- End entity: A generic term used to denote end users, devices (e.g., servers, routers).
- Certification authority (CA): The issuer of certificates and (usually) certificate revocation lists (CRLs).
- Registration authority (RA): An optional component that can assume a number of administrative functions from the CA.

● CRL issuer: An optional component that a CA can delegate to publish CRLs.

● Repository: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by End Entities.

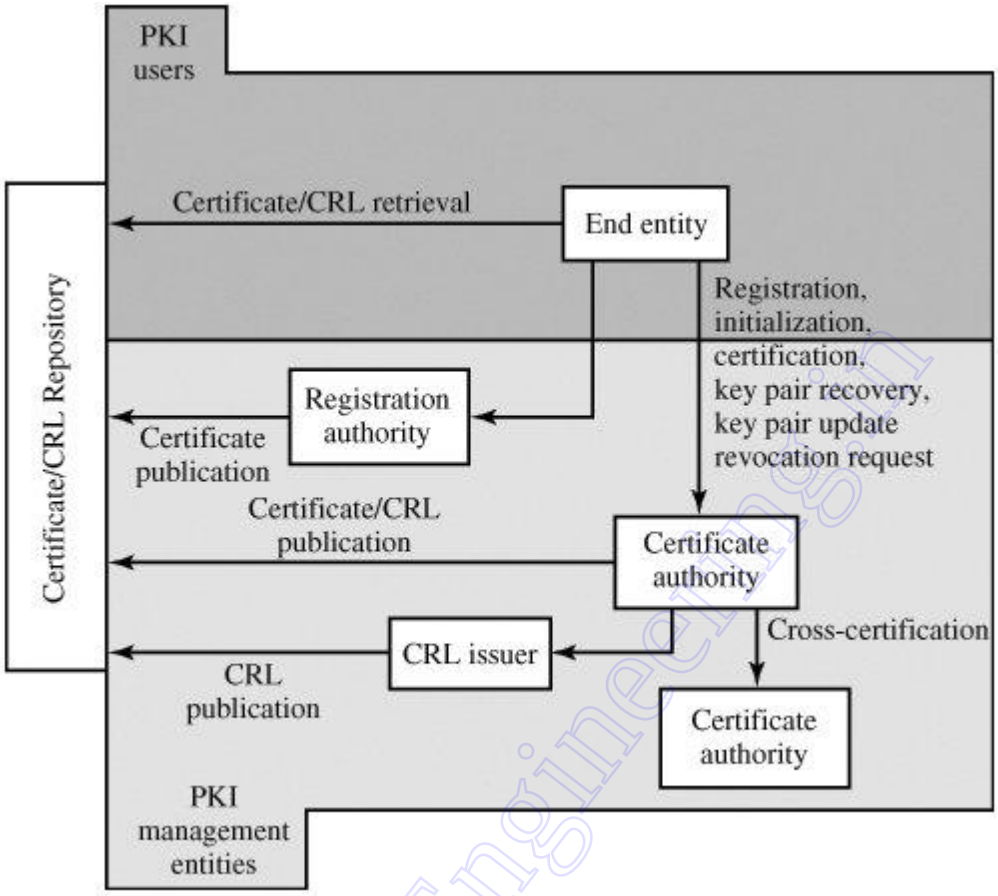


Figure: PKIX Architectural Model

PKIX Management Functions

PKIX identifies a number of management functions that potentially need to be supported by management protocols.

Registration: This is the process whereby a user first makes itself known to a CA (directly, or through an RA), prior to that CA issuing a certificate or certificates for that user. Registration begins the process of enrolling in a PKI.

Initialization: Before a client system can operate securely, it is necessary to install key materials.

For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths.

- Certification: This is the process in which a CA issues a certificate for a user's public key, and returns that certificate to the user's client system and/or posts that certificate in a repository.
- Key pair recovery: Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both.
- Key pair update: All key pairs need to be updated regularly .Update is required when the certificate lifetime expires and as a result of certificate revocation.
- Revocation request: An authorized person advises a CA of an abnormal situation requiring certificate revocation
- Cross certification: Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

PKIX Management Protocols:

The PKIX has defines two alternative management protocols between PKIX entities that support the management functions listed in the preceding subsection.

RFC 2510 defines the certificate management protocols (CMP). CMP is designed to be a flexible protocol able to accommodate a variety of technical, operational, and business models.

RFC 2797 defines certificate management messages over CMS (CMC), where CMS refers to RFC 2630, and cryptographic message syntax. CMC is built on earlier work and is intended to leverage existing implementations.

3. IP SECURITY

- **What services are provided by IP sec? (8 Marks) May/June'2007**
- **Describe about IPsecurity.(16 Marks) (IT2352 / May/June'12)**

❖ **Write on the following Apr/May'11**
Overview of IP security documents.(8 Marks)

IPSec:

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet.

Function areas of IP security:

- ✓ Authentication
- ✓ Confidentiality
- ✓ Key management

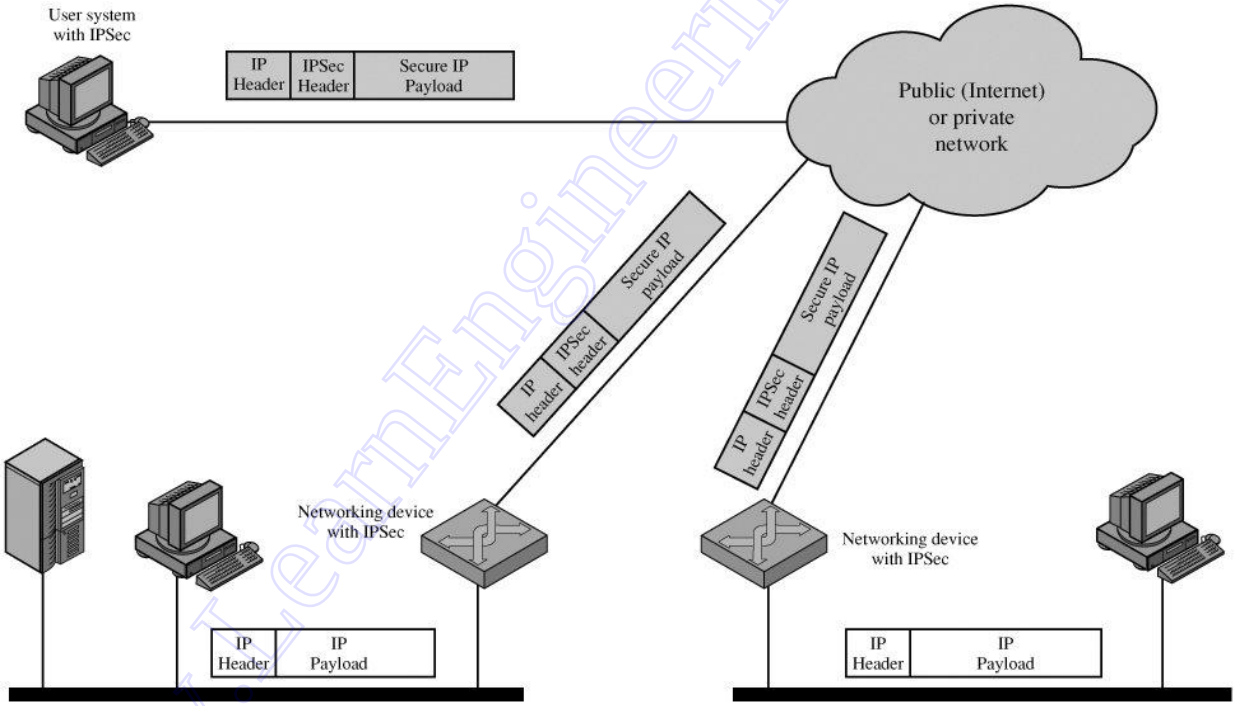


Figure: IPSecurity

Applications of IPSec.

- Secure branch office connectivity over the Internet
- Secure remote access over the Internet

- Establishing extranet and intranet connectivity with partners
- Enhancing electronic commerce security

Benefits of IPSec

- When IPSec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter.
- IPSec in a firewall is resistant to bypass if all traffic from the outside must use IP, and the firewall is the only means of entrance from the Internet into the organization.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- IPSec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
- IPSec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual sub network within an organization for sensitive applications.

Routing Applications

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

IP Security Architecture

- **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.

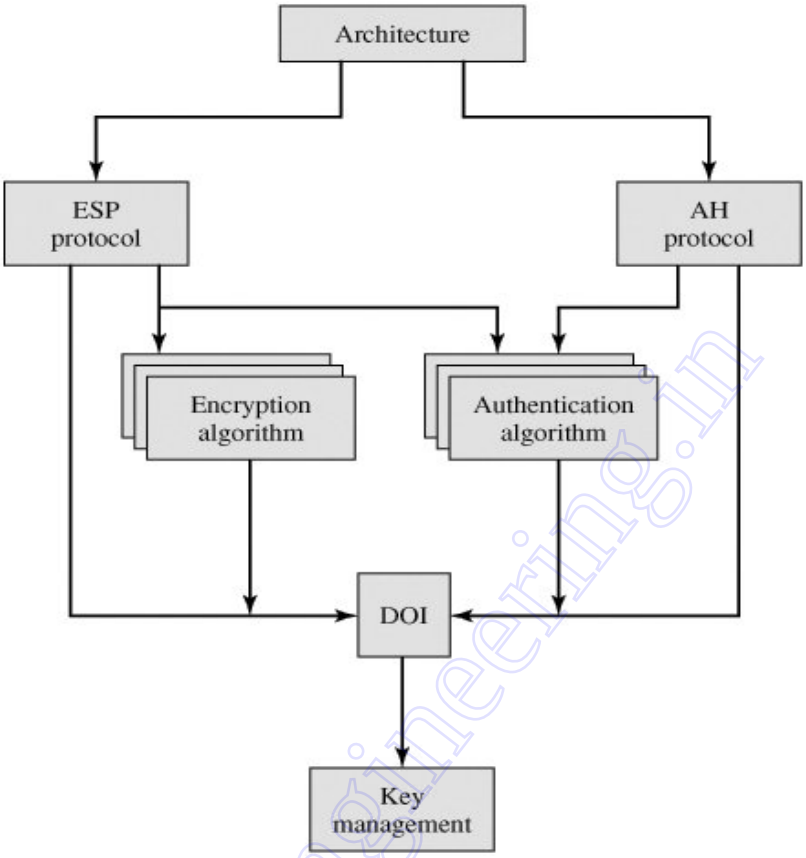


Figure: IPSec Document Overview

IPSec Services

- Access contr
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

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

Figure: IPSec Services

Authentication Header

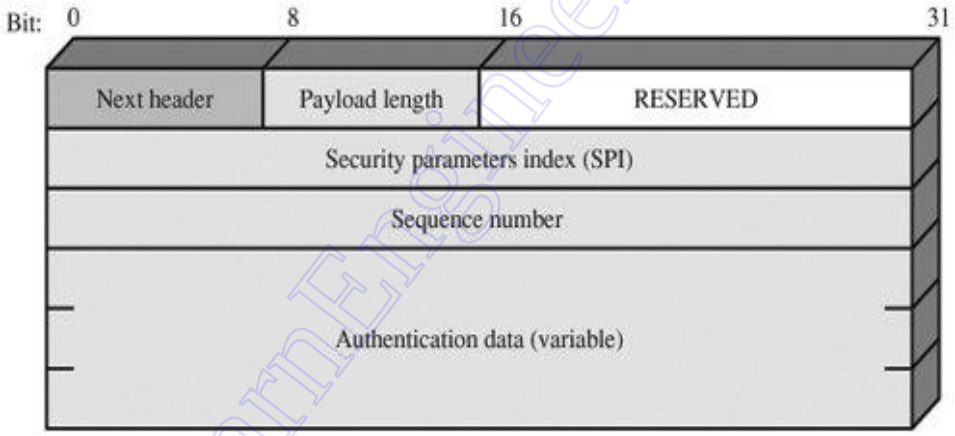


Fig 5.7 IPSec Authentication Header

Anti-Replay Service

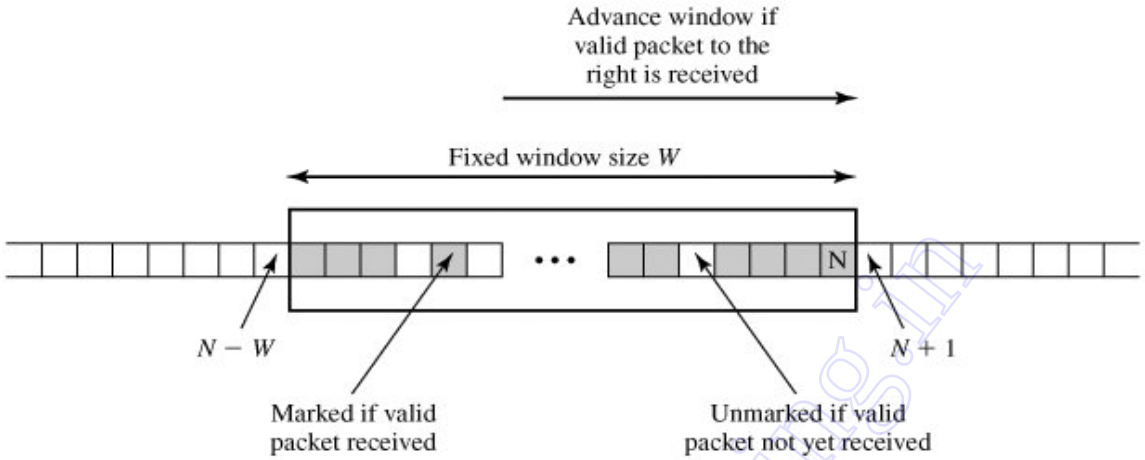


Figure: Antireplay Mechanism

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

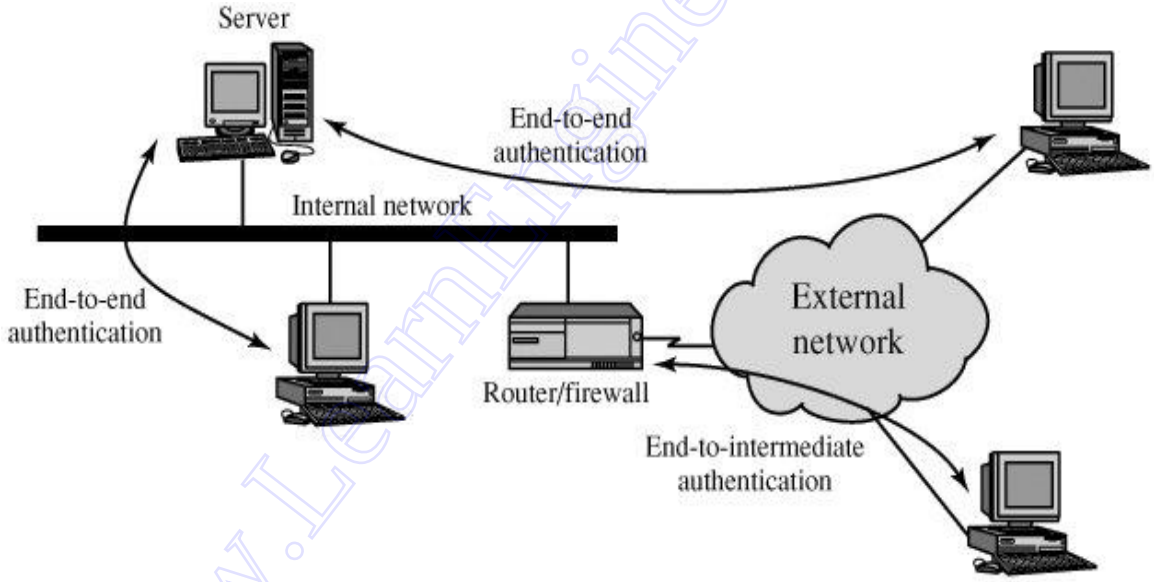


Figure: End-to-End versus End-to-Intermediate Authentication

Scope of AH Authentication

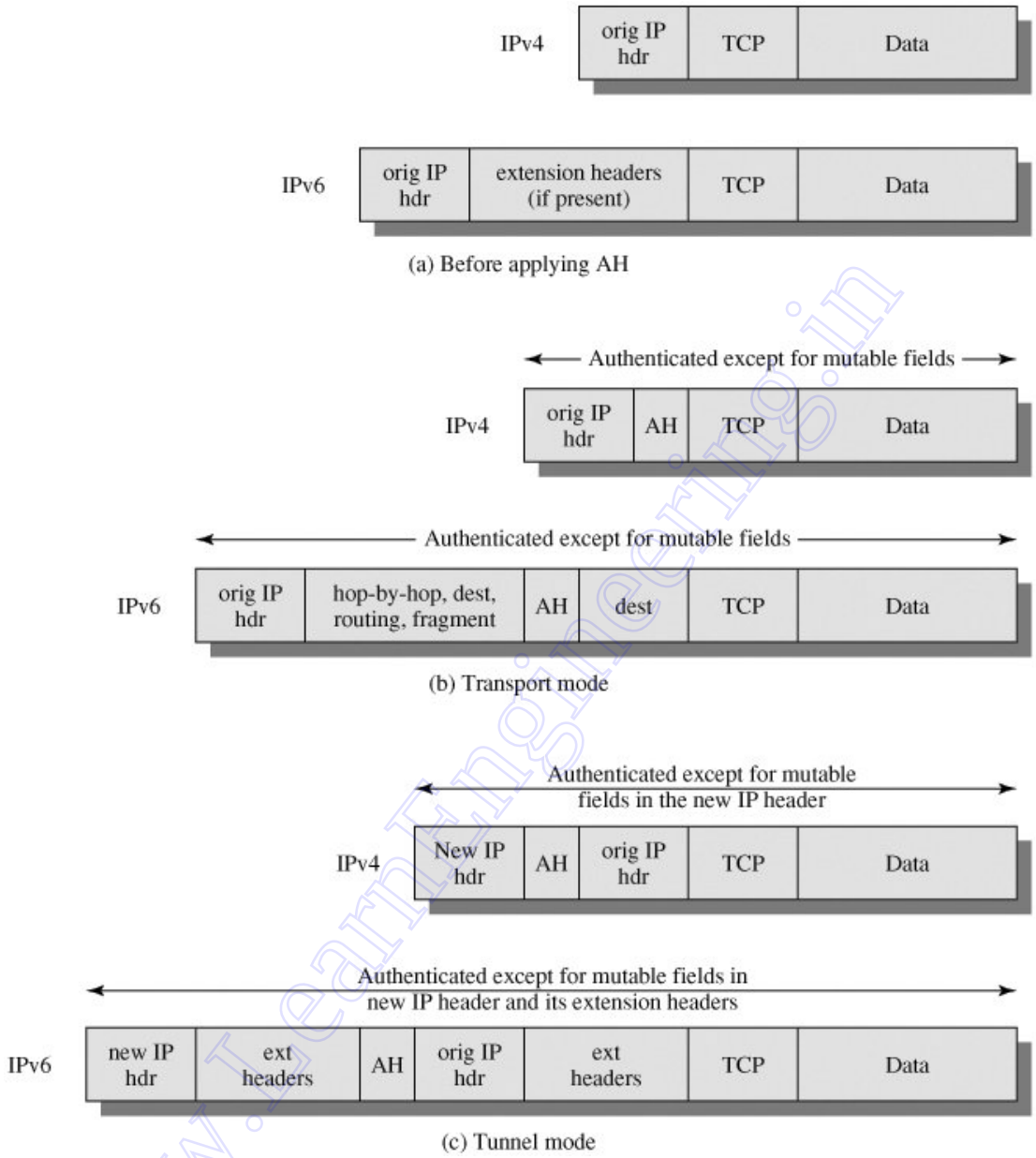


Figure: Scope of AH Authentication

4. SECURE ELECTRONIC TRANSACTION

- **List out the participants of SET system and explain in detail. (16) (IT2352 / May/June'12)**
- **Explain in detail about SET (16 Marks – Nov/Dec'12)**

Secure Electronic Transaction:

Secure Electronic Transaction (SET) is an open encryption and security specification designed to protect credit card transactions on the Internet.

- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. The current version, SETv1, emerged from a call for security standards by MasterCard and Visa in February 1996.

SET is not itself a payment system. Rather it is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network (Internet) in a secure fashion.

SET services:

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

SET Overview:

Requirements:

Business requirements for secure payment processing with credit cards over the Internet and other networks:

- Provide confidentiality of payment and ordering information: It is necessary to assure cardholders that this information is safe and accessible only to the intended recipient.

- Ensure the integrity of all transmitted data: That is, ensure that no changes in content occur during transmission of SET messages. Digital signatures are used to provide integrity.

- Provide authentication that a cardholder is a legitimate user of a credit card account. Digital signatures and certificates are used to verify that a cardholder is a legitimate user of a valid account.

- Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution.

- Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction.

- Create a protocol that neither depends on transport security mechanisms nor prevents their use.

- Facilitate and encourage interoperability among software and network providers.

Key Features of SET

SET incorporates the following features:

- Confidentiality of information.
- Integrity of data.
- Cardholder account authentication
- Merchant authentication

SET Participants:

Cardholder:

In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.

Merchant:

A merchant is a person or organization that has goods or services to sell to the cardholder.

Issuer:

This is a financial institution, such as a bank, that provides the cardholder with the payment card.

Acquirer:

This is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments. The acquirer also provides electronic transfer of payments to the merchant's account.

Payment gateway:

This is a function operated by the acquirer or a designated third party that processes merchant payment messages. The payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions.

Certification authority (CA):

This is an entity that is trusted to issue X.509v3 public-keycertificates for cardholders, merchants, and payment gateways. The success of SET will depend on the existence of a CA infrastructure available for this purpose.

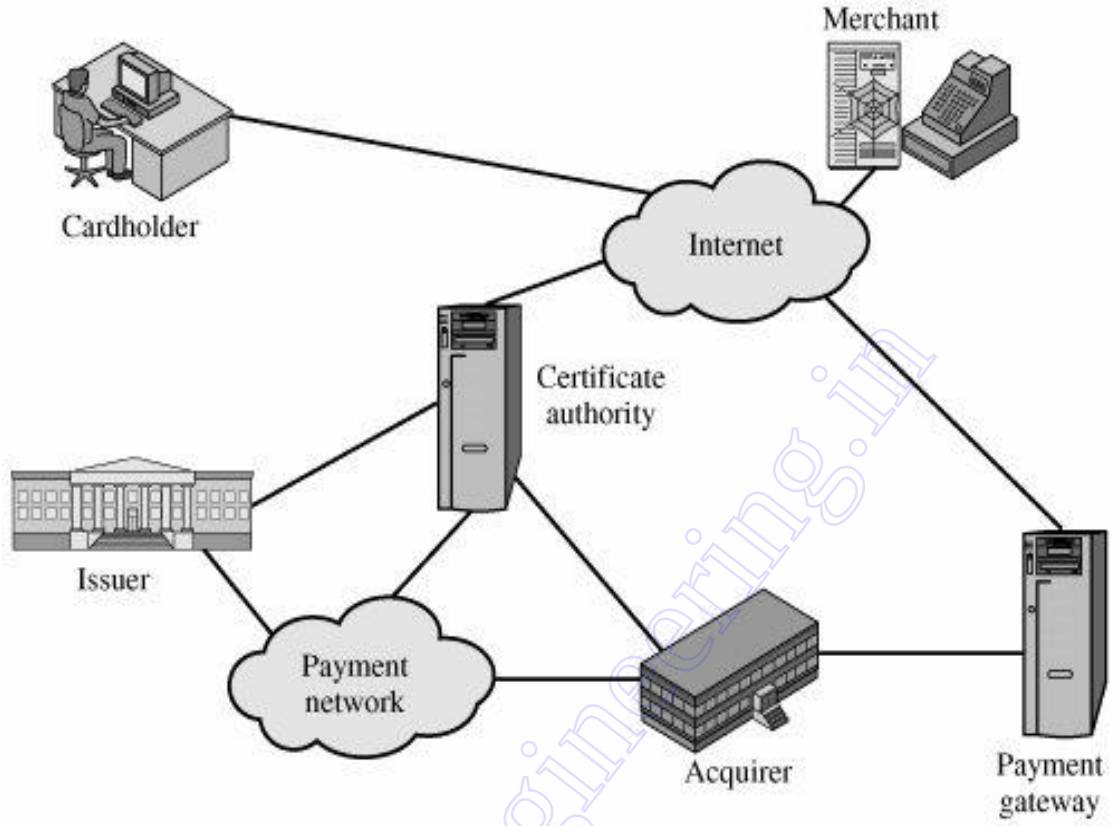


Figure: Secure Electronic Commerce Components

Sequence of events for a transaction:

- The customer opens an account
- The customer receives a certificate
- Merchants have their own certificates
- The customer places an order
- The merchant is verified.
- The order and payment are sent.
- The merchant requests payment authorization
- The merchant confirms the order.
- The merchant provides the goods or service.
- The merchant requests payment.

Dual Signature:

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

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

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

where PRc is the customer's private signature key.

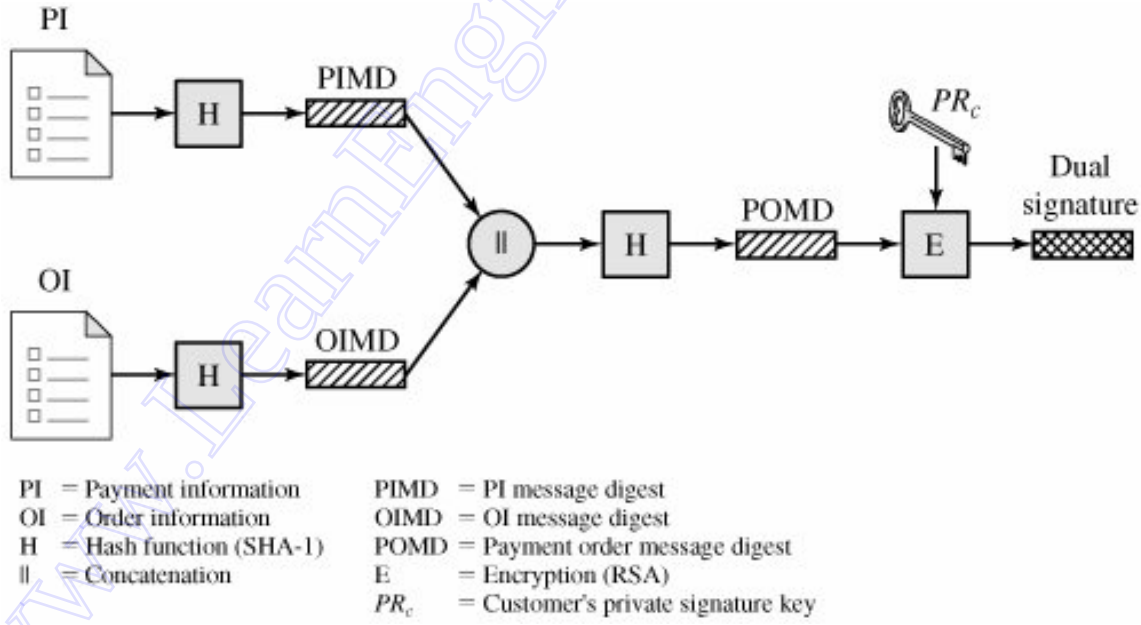


Figure: Construction of Dual Signature

if these two quantities are equal, then the bank has verified the signature.

1.The merchant has received OI and verified the signature.

2.The bank has received PI and verified the signature.

3.The customer has linked the OI and PI and can prove the linkage.

Payment Processing:

- Purchase request
- Payment authorization
- Payment capture

SET Transaction Types:

Cardholder registration:

Cardholders must register with a CA before they can send SET messages to merchants.

Merchant registration:

Merchants must register with a CA before they can exchange SET messages with customers and payment gateways.

Purchase request:

Message from customer to merchant containing OI for merchant and PI for bank.

Payment authorization:

Exchange between merchant and payment gateway to authorize a given amount for a purchase on a given credit card account.

Payment capture:

Allows the merchant to request payment from the payment gateway.

Certificate inquiry and status:

The cardholder or merchant sends the Certificate Inquiry message to determine the status of the certificate request and to receive the certificate if the request has been approved.

Purchase inquiry:

Allows the cardholder to check the status of the processing of an order after the purchase response has been received.

Authorization reversal:

Allows a merchant to correct previous authorization requests. If the order will not be completed, the merchant reverses the entire authorization.

Capture reversal :

Allows a merchant to correct errors in capture requests such as transaction amounts that were entered incorrectly by a clerk.

Credit:

Allows a merchant to issue a credit to a cardholder's account such as when goods are returned or were damaged during shipping.

Credit reversal:

Allows a merchant to correct a previously request credit. **Payment Gateway certificate request**:

Allows a merchant to query the payment gateway and receive a copy of the gateway's current key-exchange and signature certificates.

Batch administration:

Allows a merchant to communicate information to the payment gateway regarding merchant batches.

Error message:

Indicates that a responder rejects a message because it fails format or content verification tests.

5. S/MIME

- **Explain about S/MIME in detail. (16 Marks)May/June'13, Nov/Dec'12**
- **Summarize the S/MIME capability. (8 Marks) May/June'2007**

S/MIME:

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security. Both PGP and S/MIME are on an IETF standards track. S/MIME will emerge as the industry standard for commercial and or generational use, while PGP will remain the choice for personal e-mail security for many users.
S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851.

Multipurpose Internet Mail Extensions

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/UUdecode scheme. However, none of these is a standard or even a de facto standard.
2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

5. SMTP gateways to X.400 electronic mail networks cannot handle non textual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821.

Problems in S/MIME:

- Deletion, addition, or reordering of carriage return and linefeed
- Truncating or wrapping lines longer than 76 characters
- Removal of trailing white space (tab and space characters)
- Padding of lines in a message to the same length
- Conversion of tab characters into multiple space characters
- MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations.

MIME Header Fields:

The five header fields defined in MIME are as follows:

- MIME-Version: Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- Content-Type: Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- Content-Transfer-Encoding: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- Content-ID: Used to identify MIME entities uniquely in multiple contexts.
  Content-Description: A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Transfer Encodings

Table: 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.

A Multipart Example

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.

Table: Native and Canonical Form	
Native Form	The body to be transmitted is created in the system's native format.
Canonical	The entire body, including "out-of-band" information such as record

Table: Native and Canonical Form	
Native Form	The body to be transmitted is created in the system's native format.
Form	lengths and possibly file attribute information, is converted to a universal canonical form.

S/MIME provides the following functions:

- Enveloped data
- Signed data
- Clear-signed data
- Signed and enveloped data

Cryptographic Algorithms

S/MIME uses the following terminology, taken from RFC 2119 to specify the requirement level:

- **Must:** The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
- **Should:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Rules for sending agent:

1. If the sending agent has a list of preferred decrypting capabilities from an intended recipient, it SHOULD choose the first (highest preference) capability on the list that it is capable of using.

2. If the sending agent has no such list of capabilities from an intended recipient but has received one or more messages from the recipient, then the outgoing message SHOULD use the same encryption algorithm as was used on the last signed and encrypted message received from that intended recipient.

3. If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is willing to risk that the recipient may not be able to decrypt the message, then the sending agent SHOULD use tripleDES.

If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is not willing to risk that the recipient may not be able to decrypt the message, then the sending agent MUST use RC2/40.

Industrial / Practical Connectivity of the subject

Industry Connectivity and Latest Developments

Industry Connectivity

We have collaborated with EMC^2 and Infosys Pvt Limited, called Infosys campus connects. Based on the Industry requirement plan to practice the students the in cipher security and intrusion detection.

Question Paper Code : 11386

B.E / B.Tech. DEGREE EXAMINATION, APRIL/MAY 2011

Eighth Semester

Information Technology

IT2352 – CRYPTOGRAPHY AND NETWORK SECURITY

(Regulation 2008)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions

PART A – (10 * 2 = 20 Marks)

1. Differentiate passive attack from active attack with example. (Pg:6)
2. What is the use of Fermat's theorem? (Pg:6)
3. What are the different modes of operation in DES? (Pg:75)
4. Name any two methods for testing prime numbers.

   Ans: Miller-Rabin Algorithm, Deterministic primality algorithm.
5. What is discrete logarithm? (Pg:9)
6. What do you mean by one-way property in hash function? (Pg:87)
7. List out the requirements of Kerberos. (Pg:133)
8. Mention four SSL protocols. (Pg: 164)
9. Define intruder. Name three different classes of Intruders. (Pg: 132)
10. What do you mean by Trojan Horses? (Pg:132)

PART B – (5 * 16 = 80 marks)

11. (a) Discuss the classical cryptosystems and its types. (16) (Pg:10)

    (OR)

    (b) Describe Euler's and Chinese Remainder theorem. (16) (Pg: 34)

12. (a) (i) Explain about the single round of DES algorithm. (10) (Pg: 47)

    (ii) Describe key discarding process of DES. (6) (Pg: 47)

    (OR)

    (b) Explain RSA method in detail. (16) (Pg: 59)

13. (a) Discuss the discrete logarithm and explain Diffie-Hellman key Exchange algorithm with its merits and demerits. (16) (Pg: 65)

(OR)

(b) Explain about MD5 in detail. (16) (Pg: 105)

14. (a) Write on the following:

(i) Differentiate SSL from SET. (8) (Pg:172,193)

(ii) Overview of IP security documents. (8) (Pg: 185)

(OR)

(b) Explain PGP message generation and reception. (16) (Pg: 166)

15. (a) Explain definition, phases, types of virus structures and types of viruses. (16)

(Pg:144)

(OR)

(b) Write in detail about definition, characteristics, types and limitations of firewalls.

(16)(Pg:154)

Question Paper Code : 13292

B.E / B.Tech. DEGREE EXAMINATION, MAY/JUNE 2012

Eighth Semester

Information Technology

IT2352 – CRYPTOGRAPHY AND NETWORK SECURITY

(Regulation 2008)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions

PART A – (10 * 2 = 20 Marks)

1. What do you mean by **cryptanalysis**? (Pg:9)
2. What is difference between a block cipher and a stream cipher? (Pg: 43)
3. What is key distribution center? (Pg: 44)
4. Mention the application of **public key cryptography**. (Pg: 44)
5. Specify the requirements for **message authentication**. (Pg: 86)
6. What are the two important key issues related to authenticated key exchange? (Pg:88)
7. What entities constitute a full-service **Kerberos** environment? (Pg: 134)
8. Why does ESP include a padding field? (Pg:163)
9. What are the two types of audit records? (Pg: 133)
10. What is an access control matrix? What are its elements? (Pg: 160)

PART B – (5 * 16 = 80 marks)

11. (a) Explain about substitution and **transposition techniques** with two examples for each. (16)(Pg:10)

(OR)

(b) What is the need for triple **DES**? Write the disadvantages of double DES and explain **triple DES**. (16)

12. (a) Explain how the elliptic curves are useful for **cryptography**? (16) (Pg: 71)

(OR)

(b) In a public key system using **RSA**, you intercept the cipher text C=10 sent to a

user whose public key is e=5, n=35. What is the plain text? Explain the above problem with an algorithm description. (16) (Pg: 59)

13. (a) Write about the basic uses of MAC and list out the applications. (16) (Pg: 115)

(OR)

(b) with a neat sketch, explain signing and verifying functions of DSA. (16) (Pg: 126)

14. (a) Describe briefly about **X.509 authentication** procedures. And also list out the drawbacks of X.509 version 2. (16)

(OR)

(b) Write about SSL and TLS. (16) (Pg: 172)

15. (a) Explain about **intrusion detection** techniques in detail. (16) (Pg: 137)

(OR)

(b) Write about trusted systems in detail. (16) (Pg: 159)

Question Paper Code : 51560

B.E / B.Tech. DEGREE EXAMINATION, MAY/JUNE 2014

Eighth Semester

Information Technology

IT2352 – CRYPTOGRAPHY AND NETWORK SECURITY

(Regulation 2008)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions

PART A – (10 * 2 = 20 Marks)

1. What are active and passive attacks that compromise information security? (Pg:6)
2. Why random numbers are used in network security?

Ans: The uses of random numbers in cryptography are:

– nonces in authentication protocols to prevent replay

– session keys

– public key generation

– keystream for a one-time pad

3. State Euler's theorem (Pg: 7)
4. What is Optimal Asymmetric Encryption Padding?

Ans: In cryptography, Optimal Asymmetric Encryption Padding (OAEP) is a padding scheme often used together with RSA encryption. The OAEP algorithm is a form of Feistel network which uses a pair of random oracles G and H to process the plaintext prior to asymmetric encryption.

5. What is discrete logarithm problem? (Pg: 9)
6. State whether symmetric and asymmetric cryptographic algorithms need Key

Exchange. (Pg: 44)

7. List the authentication requirements. (Pg: 86)
8. What are birthday attacks? (Pg: 89)
9. Differentiate spyware and virus.

- Spyware collects information about you without appropriate notice and consent.
- A computer virus spreads software, usually malicious in nature, from computer to computer.

10. What are zombies? (Pg: 133)

PART B – (5 * 16 = 80 marks)

11. (a) Explain any two classical ciphers and also describe their security limitations.

(16)(Pg:10)

(OR)

(b) Describe Linear Feedback Shift Registers Sequences and Finite Fields with their application in cryptography. (16)

12. (a) Describe the working principle of Simple DES with an example. (16) (Pg: 47)

(OR)

(b) (i) Explain RSA algorithm. (8) (Pg: 59)

(ii) Demonstrate encryption and decryption for the RSA algorithm parameters:

p=3, q=11, e=7, d=? , M=5 (8) (Pg: 59)

13. (a) Explain Digital Signature Standard. (16) (Pg:126)

(OR)

(b) (i) Briefly explain Diffie-Hellman Key Exchange. (8) (Pg: 65)

(ii) Users A and B use the Diffie-Hellman key exchange technique with a common prime q=71 and a primitive root α=7. If user A has private key X_A=5, what is A's public key Y_A? (8) (Pg: 65)

14. (a) Elaborately explain Kerberos authentication mechanism with suitable diagrams. (16) (Pg: 134)

(OR)

(b) Explain Pretty Good Privacy in detail. (16) (Pg: 166)

15. (a) Explain statistical anomaly detection and rule based intrusion detection. (16) (Pg: 137)

(OR)

(b) Describe any two advanced anti-virus techniques in detail. (16) (Pg: 153)

Question Paper Code : 11386

B.E / B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2013

Eighth Semester

Information Technology

IT2352 – CRYPTOGRAPHY AND NETWORK SECURITY

(Regulation 2008)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions

PART A – (10 * 2 = 20 Marks)

1. Give an example each for substitution and transposition ciphers. (Pg: 8)
2. Why modular arithmetic has been used in cryptography? (Pg: 7)
3. What are the modes of DES? (74)
4. List the uses of RC4. (Pg: 46)
5. Write any two differences between MD4 and secure hash algorithm. (Pg: 88)
6. How digital signature is different from conventional? Give any two.
7. Define: SET. (Pg: 163)
8. What do you mean by PGP? (Pg: 166)
9. What are the effects of malicious software? Write any two. (Pg: 132)
10. What is 'Worm'? (Pg: 133)

PART B – (5 * 16 = 80 marks)

11. (a) (i) What are the different types of attacks? Explain. (8) (Pg: 21)

(ii) State and explain Chinese remainder theorem with example. (8) (Pg: 34)

(OR)

(b) (i) State Fermat's theorem. (4) (Pg: 29)

(ii) Fine $3^{21} \bmod 11$ using Fermat's theorem. (6) (Pg: 29)

(iii) State Euler's theorem to find gcd with example. (6) (Pg: 29)

12. (a) Write down the Triple DES algorithm and explain with neat diagram. (16)

(OR)

(b) Explain about the RSA algorithm with example as:

p=11, q=5, e=3 and PT=9 (16) (Pg: 59)

13. (a) (i) Define a hashing function. (2) (Pg: 86)

(ii) What are the properties of hashing function in cryptography? (6) (Pg: 86)

(iii) Explain Secure Hashing Algorithm (SHA). (8) (Pg: 105)

(OR)

(b) (i) Illustrate about the Birthday attacks. (8) (Pg: 104)

(ii) Explain digital signaturing with ElGamal public key cryptosystem.(8)

(Pg: 129)

14. (a) Write short notes about

(i) Web security. (8) (Pg: 172)

(ii) SSL. (8) (Pg: 174)

(OR)

(b) Explain about thr PKI. (16) (Pg: 183)

15. (a) Explain about the security standards. (16)

(OR)

(b) Write short notes on

(i) Firewalls (8) (Pg: 154)

(ii) Viruses (8) (Pg: 144)

Question Paper Code : 71763

B.E / B.Tech. DEGREE EXAMINATION, APRIL/MAY 2015

Seventh Semester

Information Technology

IT2352 – CRYPTOGRAPHY AND NETWORK SECURITY

(Regulation 2008/2010)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions

PART A – (10 * 2 = 20 Marks)

1. Differentiate between active attacks and passive attacks. (Pg: 6)
2. Find $11^7 \bmod 13$. (Pg: 7)
3. Differentiate between stream ciphers and block ciphers. (Pg: 43)
4. State few applications of RC4 algorithm. (Pg: 46)
5. What is primitive root? (Pg: 47)
6. What is digital signature? (Pg: 44)
7. What are the certificates revoked in X.509?
8. What is tunnel mode in IP Security? (Pg: 162)
9. Define worm.(Pg: 133)
10. What is the advantage of Intrusion detection system over firewall? (Pg: 137,154)

PART B – (5 * 16 = 80 marks)

11. (a) Explain the Substitution encryption techniques in detail. (16) (Pg: 10)

(OR)

(b) State and derive

(i) Fermat's theorem. (8) (Pg: 29)

(ii) Euler's theorem. (8) (Pg: 29)

12. (a) Explain Data Encryption Standard (DES) in detail. (16) (Pg: 47)

(OR)

(b) Explain about the RSA algorithm in detail. For the given values, trace the

sequence of calculations in RSA. p=7, q=13, e=5 and M=10. (16) (Pg: 59)

13. (a) Explain ElGamal public key cryptosystems with an example. (16) (Pg: 129)

(OR)

(b) Explain Secure Hash in detail. (16) (Pg: 105)

14. (a) Explain Kerberos Version 4 in detail. (16) (Pg: 134)

(OR)

(b) Explain Secure Socket Layer (SSL) in detail. (16) (Pg: 174)

15. (a) Write brief notes on the following:

(i) Classification of viruses (8) (Pg: 144)

(ii) Worm counter measures (8) (Pg: 150)

(OR)

(b) Explain the characteristics and types of firewalls. (16) (Pg: 154)

Question Paper Code : 80304

B.E / B.Tech. DEGREE EXAMINATION, NOV/DEC 2016

Seventh Semester

Computer Science and Engineering

CS6701 – CRYPTOGRAPHY AND NETWORK SECURITY

(Common to Seventh Semester Information Technology)

(Regulation 2013)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions.

PART A – (10 * 2 = 20 Marks)

1. Compare active and passive attack. (Pg:6)
2. Find gcd(1970, 1066)using Euclid's algorithm.
3. Brief the strengths of triple DES.
4. What is an elliptic curve?
5. State any three requirements for authentication.
6. Differentiate MAC and Hash function.
7. List the three classes of intruders.
8. Define Zombie.
9. List the limitations of SMTP/RFC 822.
10. Define Botnets.

PART B – (5 * 16 = 80 marks)

11. (a) (i) Explain OSI Security Architecture model with neat diagram. (8)

(ii) Describe the various security mechanisms. (8)

(OR)

(b) (i) State Chinese Remainder theorem and find X for the given set of congruent equations using CRT.

X=2(mod 3)

X=3(mod 5)

X=2(mod 7). (8)
(ii) State and prove Fermat's theorem. (8)

12. (a) Explain AES algorithm with all its round functions in detail. (16)

(OR)

(b) Explain RSA algorithm, perform encryption and decryption to the system with p=7; q=11; e=17; M=8. (Pg: 59) (16)

13. (a) Describe MD5 algorithm in detail. Compare its performance with SHA-1. (16) (Pg: 105)

(OR)

(b) Explain digital signature standard with necessary diagrams in detail. (16)

14. (a) Discuss Client Server Mutual authentication, with example flow diagram. (16)

(OR)

(b) Explain the technical details of firewall and describe any three types of firewall with neat diagram. (16)

15. (a) Discuss the working of SET with neat diagram. (16)

(OR)

(b) Explain the operational description of PGP. (16)

Question Paper Code : 71690

B.E / B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017

Seventh Semester

Computer Science and Engineering

CS 6701 – CRYPTOGRAPHY AND NETWORK SECURITY

(Common to ECE and IT)

(Regulation 2013)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL Questions

PART A – (10 * 2 = 20 Marks)

1. State Fermat's theorem. (Pg:6)
2. Determine the gcd (24140, 16762) using Euclid's algorithm. (Pg: 7)
3. State the difference between private key and public key algorithm.(Pg:44)
4. Give the five modes of operation of clock cipher.
5. What is the role of compression function in hash function?(Pg:117)
6. Specify the various types of authentication protocol.(Pg:120)
7. Define the roles of firewalls.(Pg:156)
8. State the difference between threats and attacks.(Pg:8)
9. Draw the ESP packet format.(Pg:163)
10. Specify the benefits of IPSec.(Pg:185)

PART B – (5 * 16 = 80 marks)

11. (a) State Chinese Remainder theorem and find X for the given set of congruent equations using CRT.

X=1(mod 5)
X=2(mod 7)
X=3(mod 9)
X=4(mod 11). (16)

(OR)

(b) Describe :

(i) Playfair cipher.

(ii) Railfence cipher.
(iii) Vignere cipher. (16)

12. (a) Explain Diffie-Hellman Key exchange algorithm in detail. (16)

(OR)

(b) Explain DES algorithm with neat diagram and explain the steps. (16)

13. (a) Compare the performance of RIPEMD-160 algorithm and SHA-1 algorithm. (16)

(OR)

(b) Explain the concepts of digital signature algorithm with key generation and verification in detail. (16)

14. (a) Discuss the different types of virus in detail. Suggest scenarios for deploying these types in network scenario. (16)

(OR)

(b) Explain Intrusion Detection System (IDS) in detail with suitable diagram. (16)(Pg:138)

15. (a) Discuss the architecture of IP security in detail. (16)

(OR)

(b)Discuss authentication header and ESP in detail with their packet format. (16)
