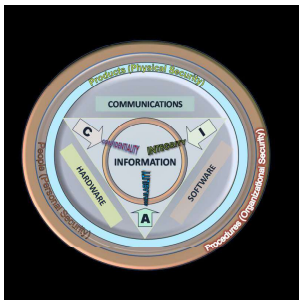
Module 1: Computer Security

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

COMPUTER SECURITY CONCEPTS

- Three concepts form what is often referred to as the **CIA triad**
- **Confidentiality:**
  - **Data confidentiality**
  - **Privacy**
- **Integrity:**
  - **Data integrity**
  - **System integrity**
- **Availability**



Basic principles

- The CIA triad of confidentiality, integrity, and availability is at the heart of information security.

Challenges of security

1: Potential Insider Threats
2: External Breaches
3. Rise of CaaS(Crime-as-a-Service)
4: Weak Links in the Supply Chain
5: IoT-centric Breaches

Challenges of Computer Security

- Mechanisms used to meet the requirements.
- Potential attacks on the security features.
- Threats
- Physical placement of security mechanisms (where to use them).
- Protocol
- Battle of wits between the attacker and the designer.
- Benefit from security investment.
- Security requires monitoring.
- Strong security and user-friendly operation of an information system

OSI security architecture

- **Architecture focuses on security attacks, mechanisms, and services**

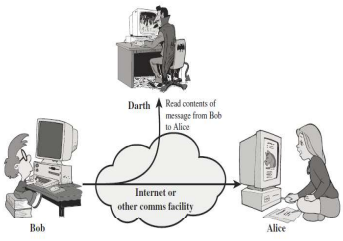
1. Security attack: Any action that compromises the security of information owned by an organization.
- *Passive attacks*
- *Active attacks*

2. Security mechanism: A process that is designed to detect, prevent, or recover from a security attack.

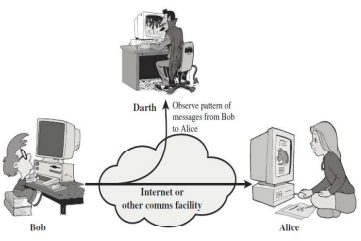
3. Security service: A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization.

1. SECURITY ATTACKS: Passive Attacks

- Release of message contents

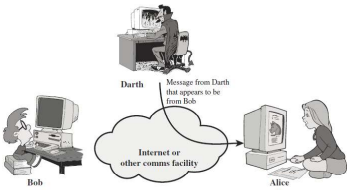


- Traffic analysis

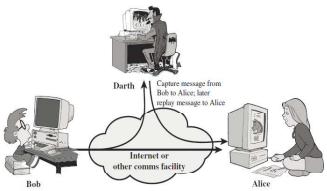


1. SECURITY ATTACKS: Active attacks

Masquerade

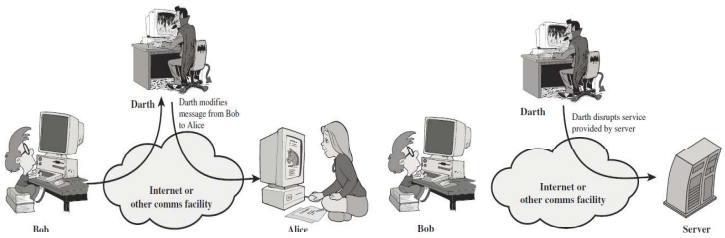


Replay



1. SECURITY ATTACKS: Active attacks

Modification of messages **Denial of service**



2. Services Mechanisms

Security mechanisms defined in X.800 are:

- **Specific security mechanisms:**
  - Encipherment, digital signature, access control, data integrity, authentication exchange, traffic padding, routing control, and notarization.
- **Pervasive Security Mechanisms:**
  - Trusted functionality, security label, event detection, security audit trail, and security recovery

3. SECURITY SERVICES

X.800 defines a security service five categories and fourteen specific services:

- **Authentication: Peer Entity Authentication and Data-Origin Authentication**
- **Access control**
- **Data confidentiality: Connection, Connectionless, Selective-Field and Traffic-Flow Confidentialities**
- **Data integrity: Connection Integrity with Recovery, Connection Integrity without Recovery, Selective-Field Connection Integrity, Connectionless Integrity and Selective-Field Connectionless Integrity**
- **Nonrepudiation: Nonrepudiation(Origin), and Nonrepudiation(Destination)**

Security policies

- A set of rules enacted by an organization to ensure that all users or networks of the IT structure within the organization's domain abide by the prescriptions regarding the security of data stored digitally within the boundaries the organization stretches its authority..

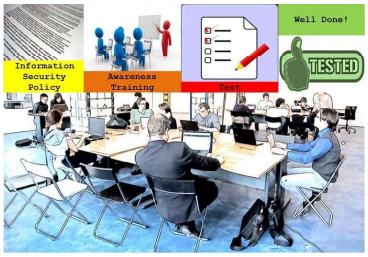
Security policies

Elements of Information Security Policy

- To establish a general approach
- To detect and forestall the compromise of information security such as misuse of data, networks, computer systems and applications.
- To protect the reputation of the company with respect to its ethical and legal responsibilities.
- To observe the rights of the customers; providing effective mechanisms for responding to complaints and queries concerning real or perceived non-compliances with the policy is one way to achieve this objective.

Information Security policies

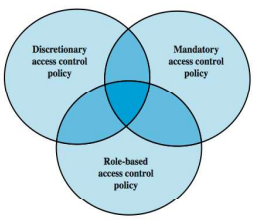
- Information security objectives
- Authority & Access Control Policy
- Classification of Data
- Data Support & Operations
- Security Awareness Sessions
- Responsibilities, Rights and Duties of Personnel



Access Control

- "The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner"

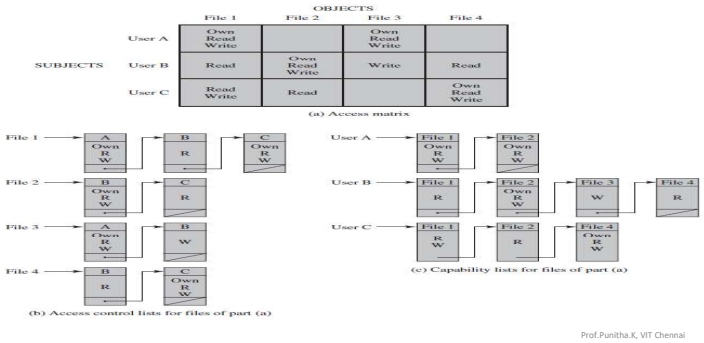
- **Access control principles:**
  - **Authentication**
  - **Authorization**
  - **Audit**



Access control structures

- Mechanisms for implementing access policies like:
- **Discretionary access control (DAC):** Access control matrix is often sparse
  - can decompose by either row (capability lists) or column (Access control lists - ACLs)
  - **Mandatory access control (MAC)**
  - **Role-based access control (RBAC)**
  - Intermediate controls (groups, negative permissions, roles, protection rings etc.)
- Requirements for access control structures: an ability to express control policies verifiability of correctness. scalability and manageability

Access Control Structures



Module 2:

What is block cipher?

- In cryptography, a block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key.
- A block cipher algorithm is a basic building block for providing data security.
- To apply a block cipher in a variety of applications, five "modes of operations" are defined by NIST.

Introduction to block cipher modes of operation

- NIST : National Institute of Standards and Technology
- That five modes of operations are :

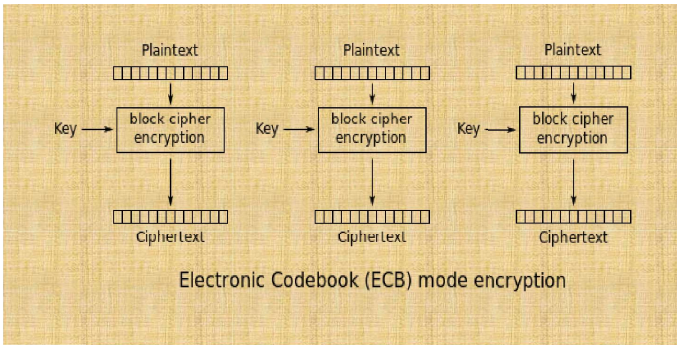
1. Electronic code book
2. Cipher chaining block
3. Cipher feedback mode
4. Output feedback mode
5. Counter mode

When we use block cipher modes of operation?

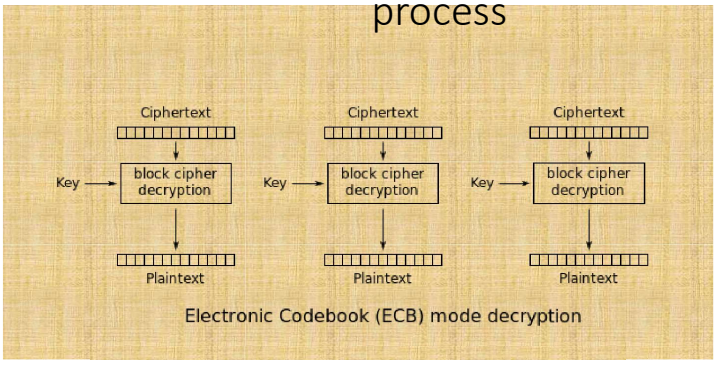
- Block cipher only allow to encrypt entire blocks.
- What if our message is longer/shorter than the block size?
- When message is longer/shorter than the block size , we use modes of operations.
- Algorithms that exploit a block cipher to provide a service (e.g. confidentiality).

Electronic code book

- ECB is the simplest mode of operation.
- The plain text is divided into N blocks.
- The block size is n bits.
- If the plaintext size is not multiple of the block size , the text is padded to make the last block the same size other blocks.
- Same key is used to encrypt and decrypt each block.



process



Electronic codebook encryption/decryption Security issues

- Patterns at the block level are preserved.
- For example equal blocks in the plain text become equal block in the cipher text.
- If any person finds out the cipher text block 1,5 and 10 are the same ,that person knows that plaintext blocks 1, 5 and 10 are the same.
- This is a leak in security.

What is initialization vector?

- An initialization vector (IV) or starting variable is a block of bits that is used by several modes to randomize the encryption and hence to produce distinct cipher texts even if the same plain text is encrypted multiple times, without the need for a slower re-keying process.
- An initialization vector has different security requirements than a key, so the IV usually does not need to be secret
- However, in most cases, it is important that an initialization vector is never reused under the same key.

What is initialization vector? (continue…)

- For CBC and CFB, reusing an IV leaks some information about the first block of plaintext, and about any common prefix shared by the two messages.
- For OFB and CTR, reusing an IV completely destroys security. This can be seen because both modes effectively create a bit stream that is XORed with the plaintext, and this bit stream is dependent on the password and IV only. Reusing a bit stream destroys security.
- In CBC mode, the IV must, in addition, be unpredictable at encryption time; in particular, the (previously) common practice of re-using the last cipher text block of a message as the IV for the next message is insecure.

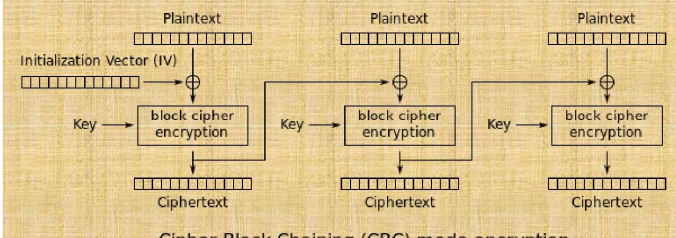
Cipher block chaining mode

- IBM invented the Cipher Block Chaining (CBC) mode of operation in 1976.
- In CBC mode, each block of plaintext is XORed with the previous cipher text block before being encrypted.
- This way, each cipher text block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.

Cipher block chaining mode encryption

IV = initialization vector

Cipher text$_i$ = encryption with key (plain text XOR cipher text $_{i-1}$)

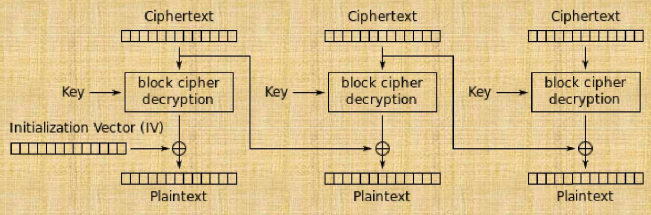


Cipher Block Chaining (CBC) mode encryption

Cipher block chaining mode Decryption

IV = initialization vector

plain text$_i$ = Decryption with key (cipher text XOR cipher text $_{i-1}$)



Cipher Block Chaining (CBC) mode decryption

Cipher block chaining mode Security issues

- The patterns at the block level are not preserved.
- In CBC mode, equal plain text block belonging to the same message are enciphered into different cipher text block.
- However ,if two message are equal ,their encipherment is the same if they use the same IV.
- As a matter of fact ,if the first M blocks in two different message are equal , they are enciphered into equal blocks unless different IVs are used.
- For this reason , some people recommended the use of time-stamp as an IV.
- Any person can add some cipher text blocks to the end of the cipher text stream.

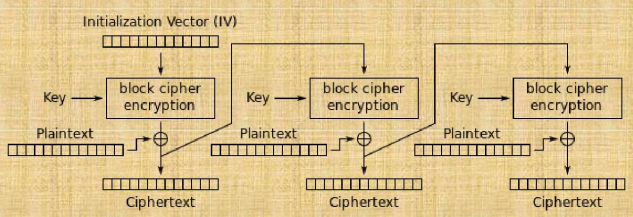
Cipher feedback mode

- ECB and CBC modes encrypt and decrypt blocks of the message.
- Block size n is predetermine by the underlying cipher ; for example , for DES n = 64
  for AES n =128
- In some situations, we need use DES or AES as secure cipher , but the plain text or cipher text block size are to be smaller.
- For example , to encrypt and decrypt 8-bit characters , you would not want to use one of the traditional cipher like Caesar cipher.
- The solution is to use DES or AES in cipher feedback mode

Cipher feedback mode encryption

IV = initialization vector

Cipher text$_i$ = Encryption with key (Cipher text $_{i-1}$) XOR plain text

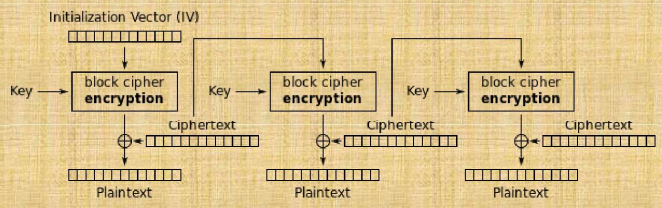


Cipher Feedback (CFB) mode encryption

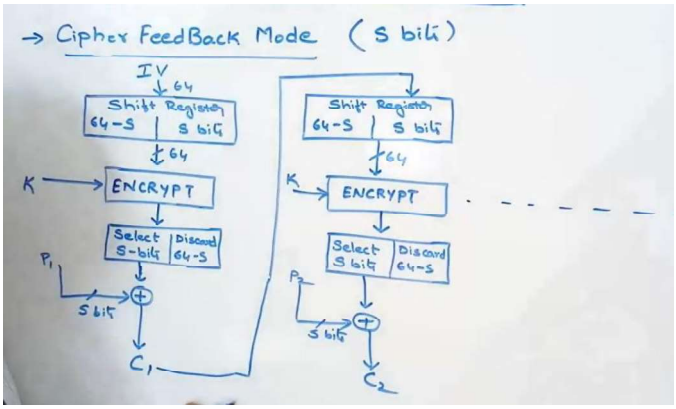
Cipher feedback mode Decryption

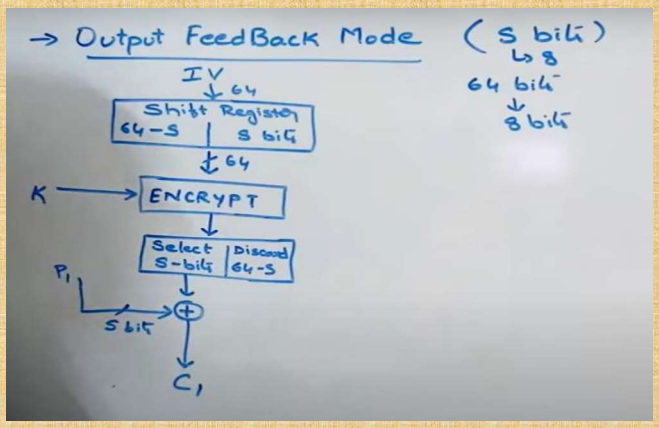
IV = initialization vector

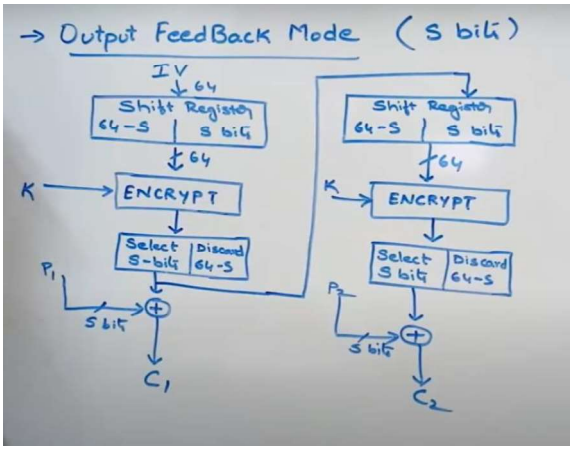
Plain $text_i$ = Encryption with key (Cipher $text_{i-1}$) XOR cipher text



Cipher Feedback (CFB) mode decryption







Cipher feedback mode Security issues

- Just like CBC , patterns at the block level are not preserved.
- More than one message can be encrypted with the same key , but the value of the IV should be changed for each message.
- This means that sender needs to use a different IV each time sender sends a message.
- Attacker can add some cipher text block to the end of the cipher text stream.

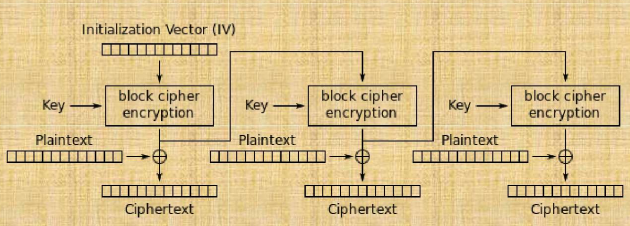
Cipher output feedback mode

- Output feedback mode is very similar to CFB mode , with one difference: each bit in the cipher text is independent of the previous bit or bits.
- This avoids error propagation.
- If an error occur in transmission , it does not affect the bits that follow.
- Note that , like cipher feedback mode , both the sender and the receiver use the encryption algorithm.

output feedback mode encryption

IV = initialization vector cipher $text_i$ = plain $text_i$ XOR Encryption

(k , [cipher text i-1 XOR plain $text_{i-1}$])



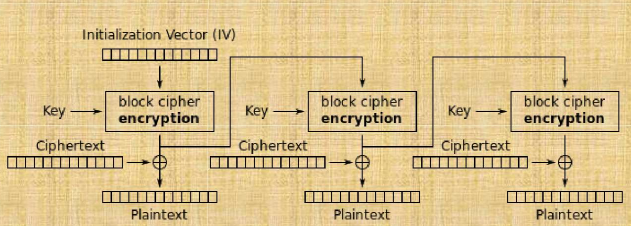
Output Feedback (OFB) mode encryption

Cipher output feedback mode Decryption

IV = initialization vector

Plain $text_i$ = cipher $text_i$ XOR Encryption

(k , [cipher text i-1 XOR plain $text_{i-1}$])



Output Feedback (OFB) mode decryption

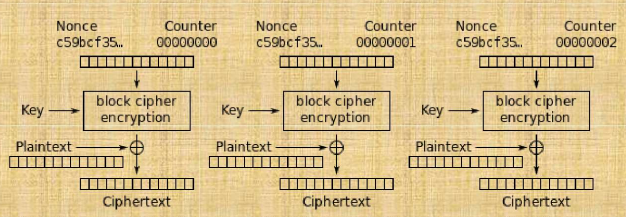
Counter

- In the counter mode , there is no feedback.
- The pseudo randomness in the key streams achieved using a counter.
- An n bit counter is initialized to a predetermined value(IV) and incremented based on a predefined rule(mod 2^n)
- To provide a better randomness , the increment value can depend on the block numbers to be incremented.
- The plain text and cipher block text block have same block size as the underlying cipher.
- Both encryption and decryption can be performed fully in parallel on multiple blocks .
- Provides true random access to cipher text blocks

Counter Encryption process

Nonce = IV

Cipher texti = Plain texti XOR Encryption with key(counter)

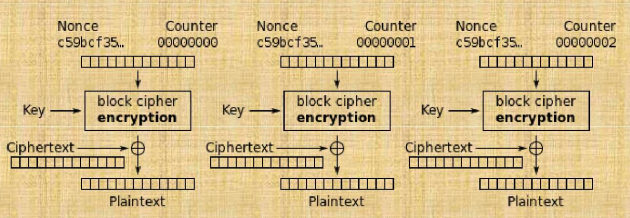


Counter (CTR) mode encryption

Counter Decryption process

Nonce = IV

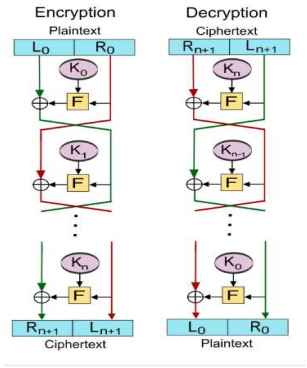
Plain texti = Cipher texti XOR Encryption with key(counter)



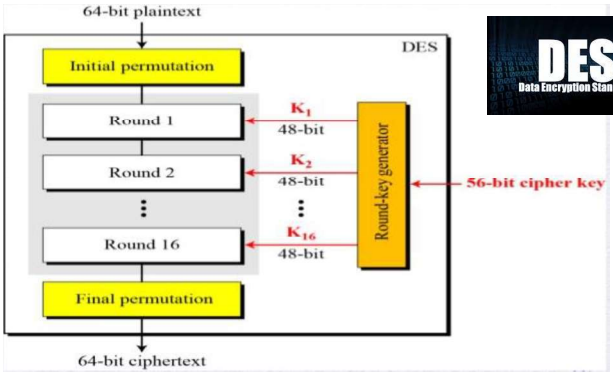
Counter (CTR) mode decryption

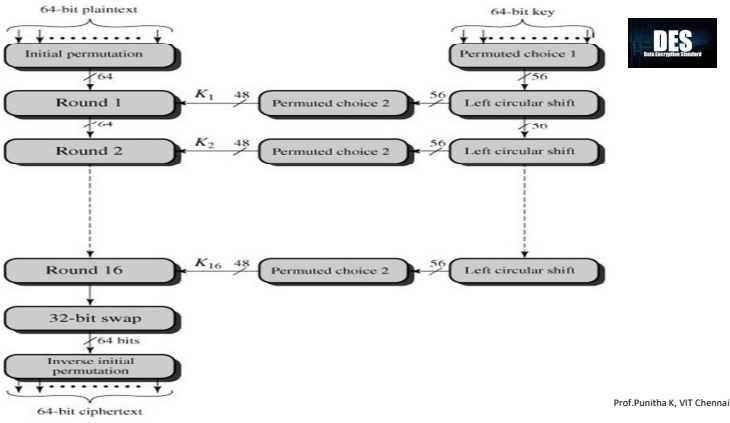
- Plain text – 64 bits
- Key size – 64 bits
- 16 rounds
- Sub keys – 16 sub keys (each sub key size – 48 bits)
- Permutation (transposition cipher)
- Feistel structure
- Round function
- Substitution method

Module 2



Feistel Structure

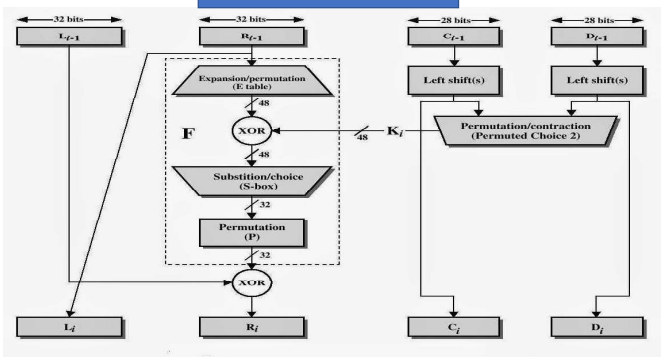




Initial Permutation

Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Round Function of DES



Prof.Punitha K, VIT Chennai

Expansion Permutation Table

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

Permuted Choice1 (PC1)

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

Bits Rotation

- Before the round sub-key is selected, each half of the key schedule state is rotated left by a number of places. This table specifies the number of places rotated.
- • The key is divided into two 28-bit parts
- • Each part is shifted left (circular) one or two bits
- in

Bits Rotation Table

Number of Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left rotations	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

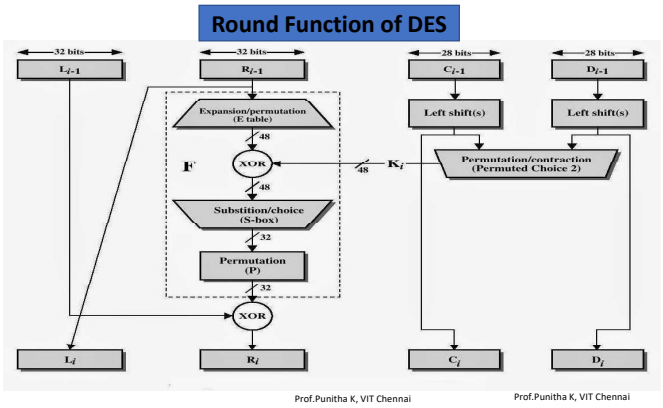
Permuted choice 2 (PC-2)

- This permutation selects the 48-bit sub-key for each round from the 56-bit key-schedule state. This permutation will ignore 8 bits below:

Permuted Choice 2 "PC-2" Ignored bits 9,18,22,25,35,38,43,54.

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

Round Function of DES

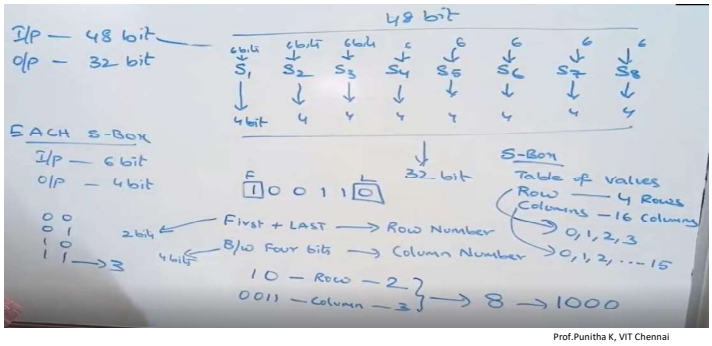


Prof.Punitha K, VIT Chennai

S box in DES

- Input – 48 bits, Output – 32 bits, Reduction is done by using '8' S box from 48 input bits as S1, S2, S3, S4, S5, S6, S7, S8
- Each S-box replaces a 6-bit input with a 4-bit output.
- Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits, and the column using the inner four bits.
- For example, an input "011011" has outer bits "**01**" and inner bits "1101"; noting that the first row is "00" and the first column is "0000", the corresponding output for S-box S_5 would be "1001" (=9), the value in the second row, 14th column.
- S box consist of 4 rows (0,1,2,3)and 16 columns (0,1…..15)

S box in DES



S box

S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Permutation (P)

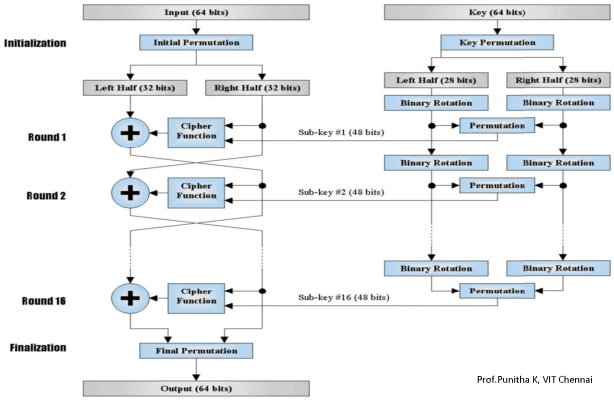
- The P permutation shuffles the bits of a 32-bit half-block

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

32 bit swap then apply Final Permutation (IP^{-1})

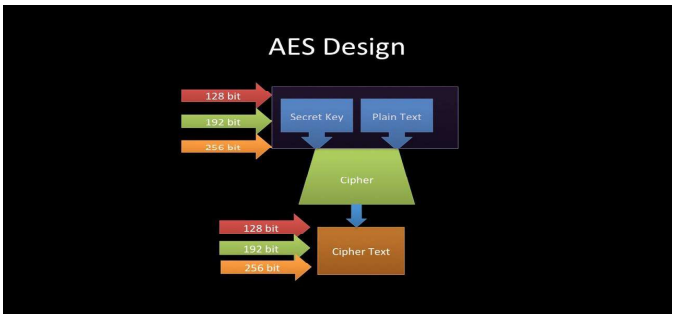
Final Permutation							
40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25



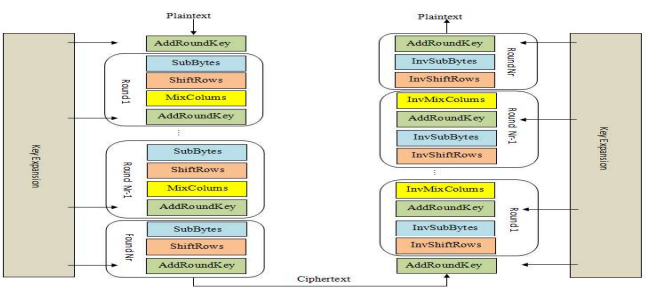
Advanced Encryption Standard

- Operation of AES
- Encryption Process
- Decryption Process
- AES Analysis

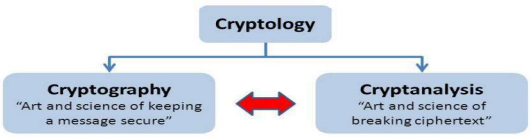
Advanced Encryption Standard



Advanced Encryption Standard



What is Cryptology?

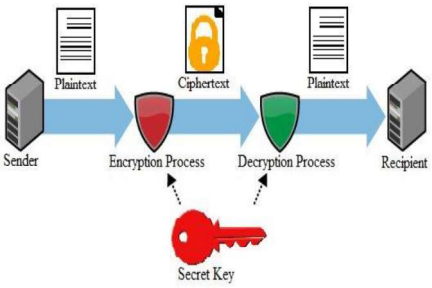


- Cryptology covers two related fields:
  - Cryptography: how to keep a message secure (develop ciphers that are unbreakable)
  - Cryptanalysis: how break ciphers and cipher-text

- The study and application of techniques that hide the real meaning of information by transforming it into nonhuman readable formats and vice versa.
- A method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. The prefix "crypt-" means "hidden" or "vault" -- and the suffix "-graphy" stands for "writing."

Components of a Cryptosystem

- **Sender**
- **Recipient**
- **Plaintext**
- **Encryption Algorithm**
  - produces a ciphertext
- **Ciphertext**
- **Decryption Algorithm**
  - to compute the ciphertext
- **Encryption Key**
- **Decryption Key**



Classical encryption techniques

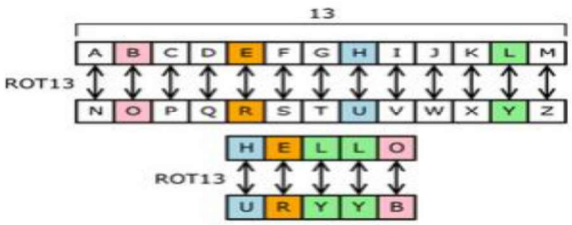
Substitution
Letters of plain text are replaced by other letters or by numbers or symbols

Transposition/Permutation
Letters/bytes/bits of the plaintext are rearranged without altering the actual letters used

Substitution Techniques

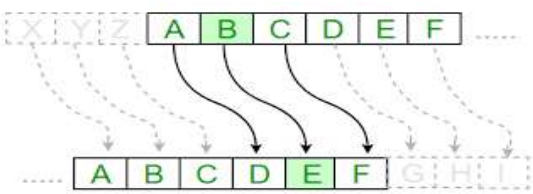
- Simple substitution
- Caesar cipher
- Monoalphabetic Cipher
- Polyalphabetic Cipher
  - One Pad cipher
  - Playfair Cipher

Simple substitution



Caesar cipher

- Key



Monoalphabetic Substitutions

- Monoalphabetic Cipher – always uses the same letter of the alphabet for the ciphertext letter.

A – 0	H – 7	O – 14	V – 21
B – 1	I – 8	P – 15	W – 22
C – 2	J – 9	Q – 16	X – 23
D – 3	K – 10	R – 17	Y – 24
E – 4	L – 11	S – 18	Z – 25
F – 5	M – 12	T – 19	
G – 6	N – 13	U – 20	

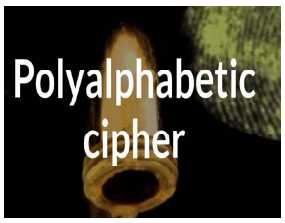
- Arithmetic is done as if the alphabet table were circular.
- Encrypt: $C = E(a) = (a + k) \bmod 26$
- Decrypt: $a = D(C) = (C - k) \bmod 26$

One time Pad Cipher

Example2: Ciphertext: " UQXTQUYFR " Key = NCBTZQARX Plaintext: ?

Ciphertext	U	Q	X	T	Q	U	Y	F	R
Cipher Code	20	16	23	19	16	20	24	5	17
Key	N	C	B	T	Z	Q	A	R	X
Kencoding	13	2	1	19	25	16	0	17	23
Pencoding	7	14	22	0	17	4	24	14	20
Plaintext	H	O	W	A	R	E	Y	O	U

16 – 25 = – 9 26 – 9 = 17



Playfair Cipher:

Key:MONARCHY,Plaintext:"instruments"

Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'

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

in:

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

st:

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

ru:

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

me:

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

nt:

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

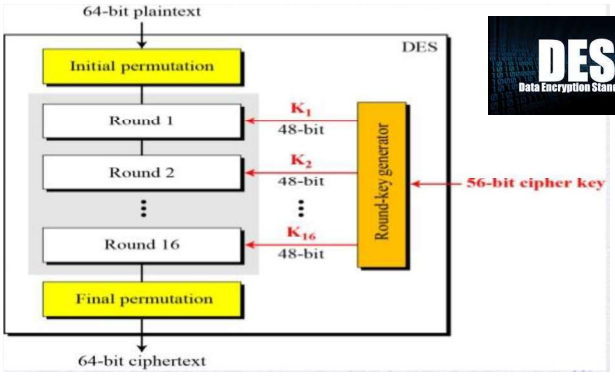
sz:

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


```
Key:          4 3 1 2 5 6 7
Plaintext:    a t t a c k p
              o s t p o n e
              d u n t i l t
              w o a m x y z
Ciphertext:   TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

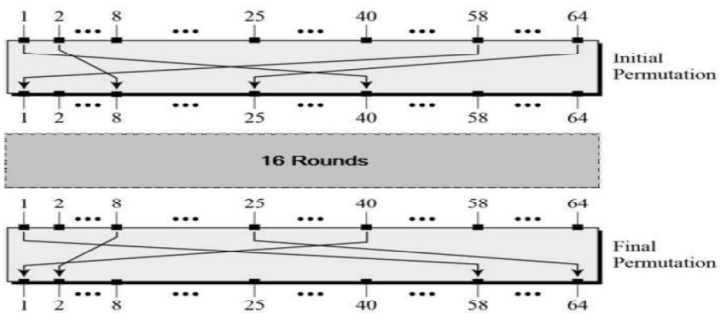

Block ciphers

- Block Size
  - Avoid very small block size, Do not have very large block size, Multiples of 8 bit
- Padding in Block Cipher
  - Blocks of fixed sizes (say 64 bits)
  - Eg. 150-bit plaintext provides two blocks of 64 bits each with third block of balance 22 bits. The last block of bits needs to be padded up with redundant bits padding.
- Block Cipher Schemes
  - Digital Encryption Standard (DES), Triple DES, Advanced Encryption Standard (AES), IDEA, Twofish, Serpent

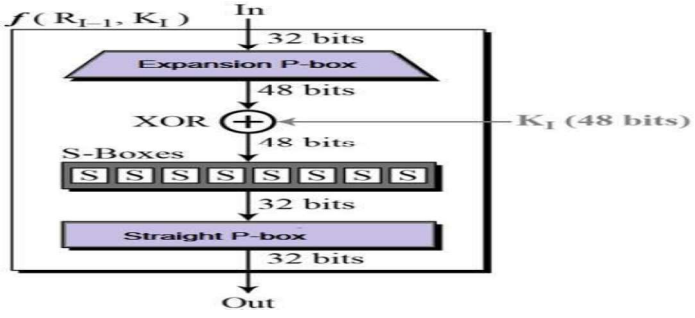


- Initial and final permutation
- Round function
- Key schedule

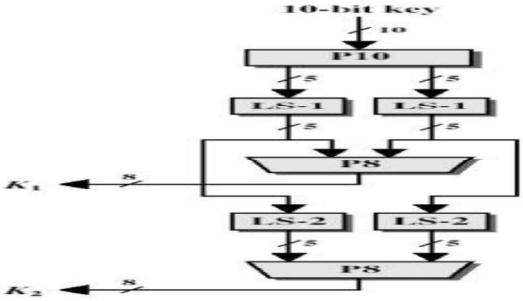
Initial and final permutation in DES



Round Function in DES



Key generation in DES



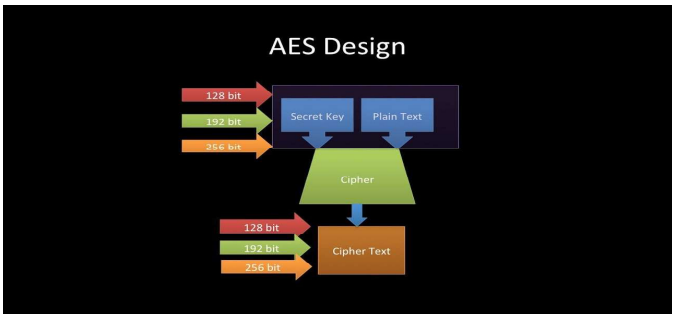
DES Analysis

- Two properties make cipher very strong.
- **Avalanche effect** – A small change in plaintext results in the very great change in the ciphertext.
- **Completeness** – Each bit of ciphertext depends on many bits of plaintext.

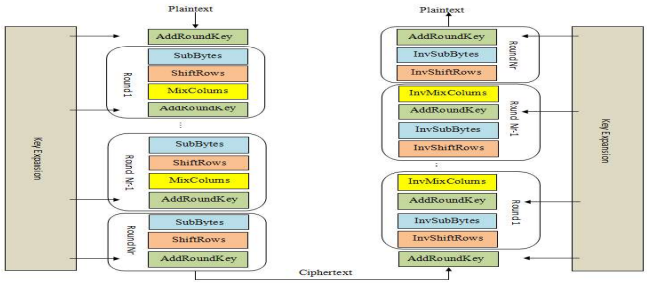
Advanced Encryption Standard

- Operation of AES
- Encryption Process
- Decryption Process
- AES Analysis

Advanced Encryption Standard



Advanced Encryption Standard



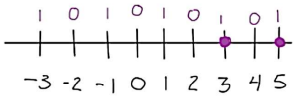
Module 3: Modular Arithmetic

$3 \equiv 5 \pmod 2$

$\text{mod } 2 \quad \{0, 1\}$

$\text{mod } 3 \quad \{0, 1, 2\}$

$\text{mod } 9 \quad \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$



$3 \equiv 5 \pmod 2$

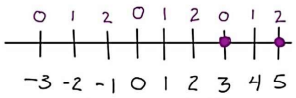
$\text{mod } 2 \quad \{0, 1\}$

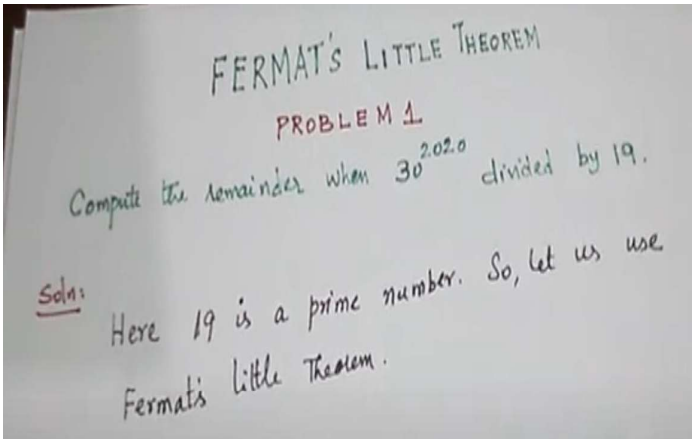
$\text{mod } 3 \quad \{0, 1, 2\}$

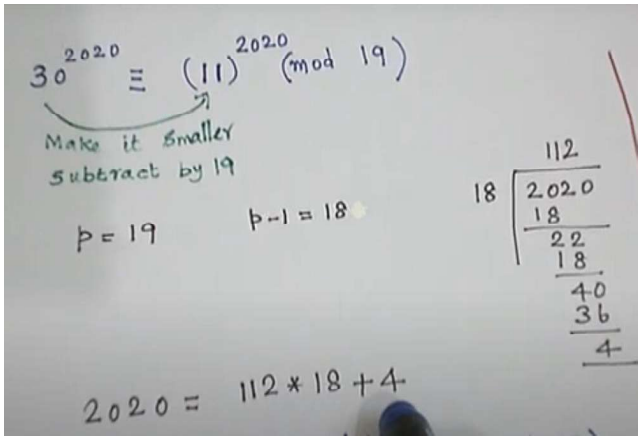
$3 \not\equiv 5 \pmod 3$

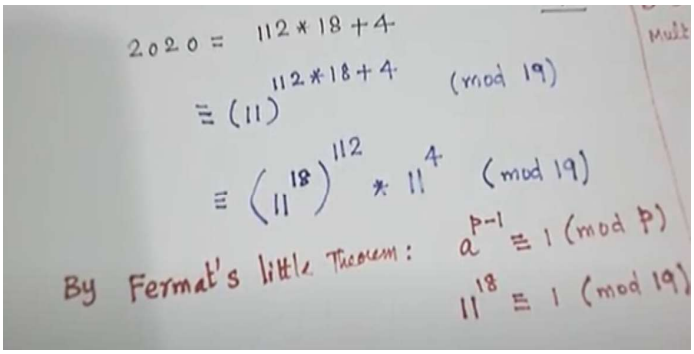
$3 \equiv 0 \pmod 3$

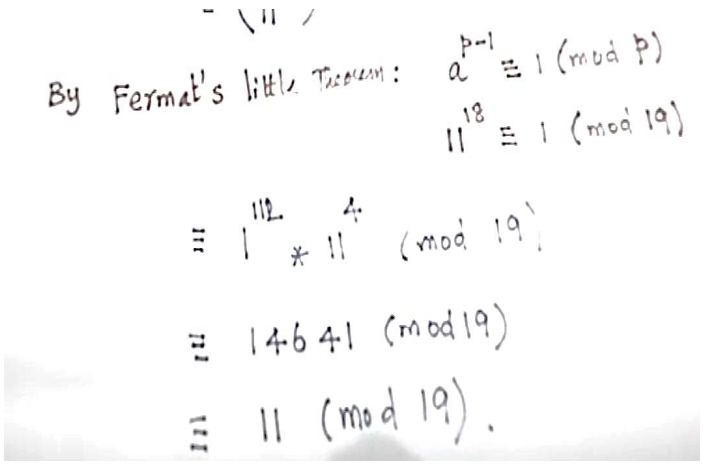
$\text{mod } 9 \quad \{0, 1, 2, 3, 4, 5, 6, 7, 8\} \quad 5 \equiv 2 \pmod 3$











Module 3 :Diffie-Hellman key exchange

- Not an Encryption algorithm
- Exchange of Secret/ symmetric key
- Asymmetric encryption

What is Diffie-Hellman?

- A Public Key Algorithm
- Only for Key Exchange
- Does NOT Encrypt or Decrypt
- Based on Discrete Logarithms
- Widely used in Security Protocols and Commercial Products
- Williamson of Britain's CESG claims to have discovered it several years prior to 1976

Discrete Logarithm Problem

- The security of the Diffie-Hellman algorithm depends on the difficulty of solving the discrete logarithm problem (DLP) in the multiplicative group of a finite field

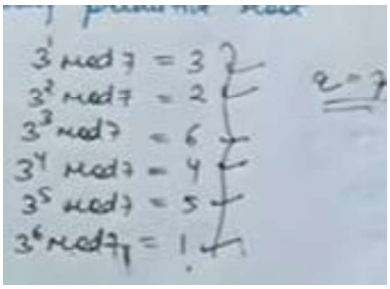
Diffie-Hellman Algorithm

- Five Parts

1. Global Public Elements
2. User A Key Generation
3. User B Key Generation
4. Generation of Secret Key by User A
5. Generation of Secret Key by User B

Global Public Elements

- q Prime number
- α, $\alpha < q$ and α is a primitive root of q
- The global public elements are also sometimes called the domain parameters



User A Key Generation

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

User B Key Generation

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

Generation of Secret Key by User A, B

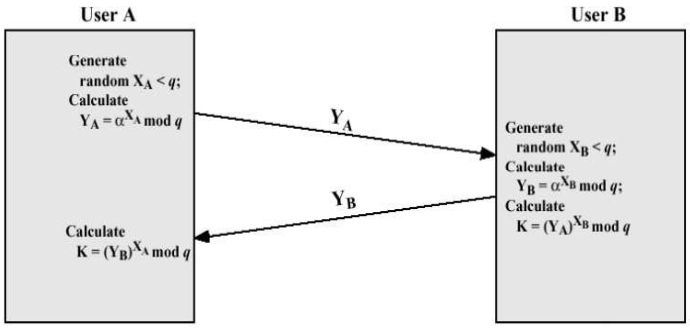
Generation of Secret Key by User A

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

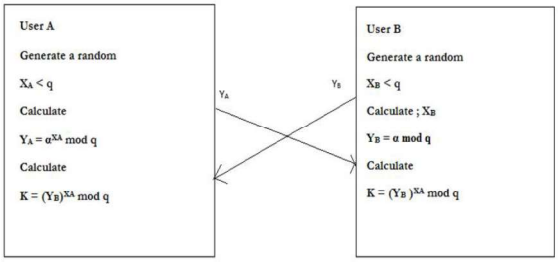
Generation of Secret Key by User B

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

Diffie-Hellman Key Exchange



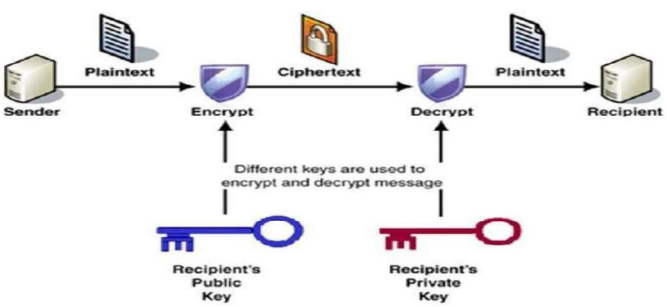
Diffie-Hellman key exchange



Principles of Public-Key Cryptosystems

- The concept of public key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.
- Key distribution under symmetric key encryption requires either
  - (1) that two communicants already share a key, which someone has been distributed to them or
  - (2) the use of a key distribution center.
- Digital signatures.

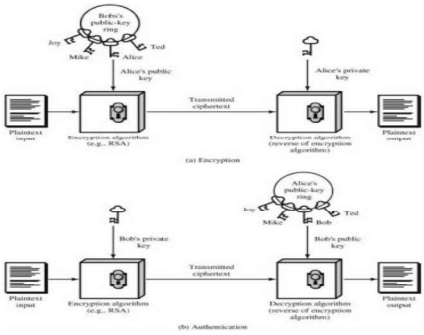
Principles of Public-Key Cryptosystems



Principles of Public-Key Cryptosystems

- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.
- Characteristic: Computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
- Either of the two related keys can be used for encryption, with the other used for decryption.
- A public-key encryption scheme has six ingredients : Plain text, encryption algorithm, public and private keys, cipher text, decryption algorithm.

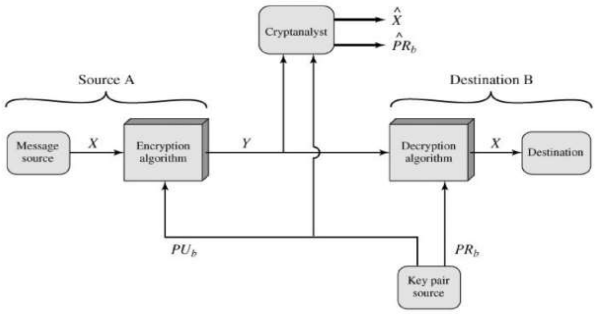
Principles of Public-Key Cryptosystems



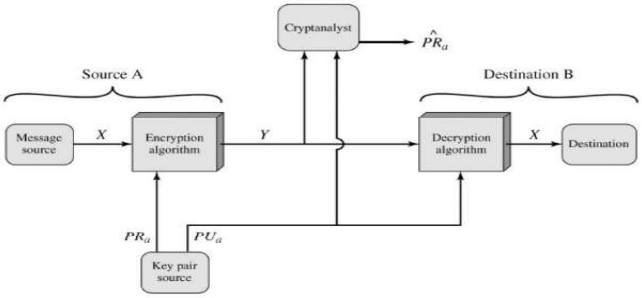
Principles of Public-Key Cryptosystems

Conventional Encryption	Public-Key Encryption
Needed to Work: The same algorithm with the same key is used for encryption and decryption. The sender and receiver must share the algorithm and the key.	**Needed to Work:** One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security: Needed The key must be kept secret. It must be impossible or at least impractical to decipher a message if no other information is available. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	**Needed for Security: Needed** One of the two keys must be kept secret. It must be impossible or at least impractical to decipher a message if no other information is available. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

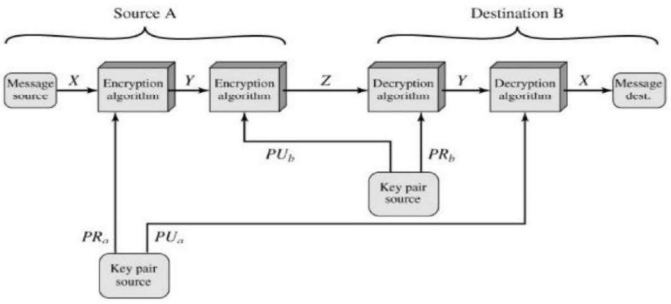
Principles of Public-Key Cryptosystems



Principles of Public-Key Cryptosystems



Principles of Public-Key Cryptosystems



Principles of Public-Key Cryptosystems

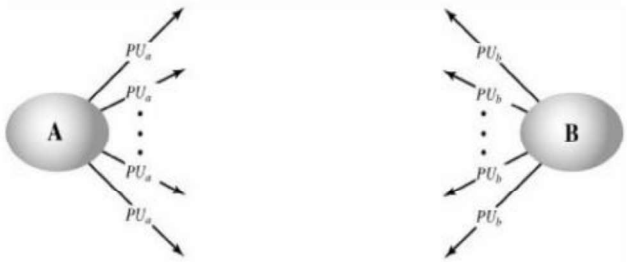
Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
Key exchange: Two sides cooperate to exchange a session key. Several different approaches are possible,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

Principles of Public-Key Cryptosystems

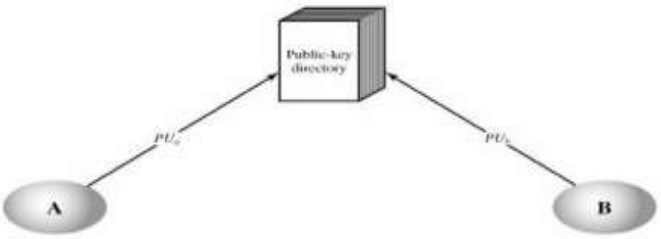
Key Management:
Distribution of Public Keys:

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

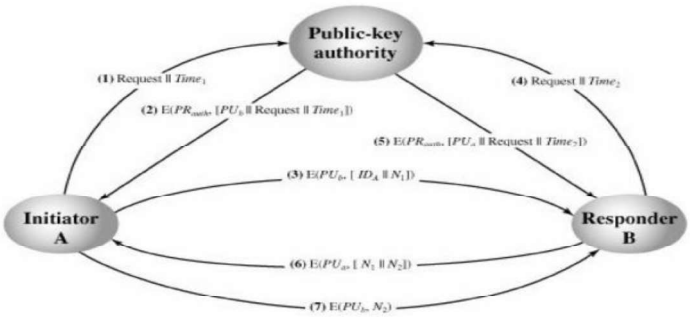
Public Announcement of Public Keys



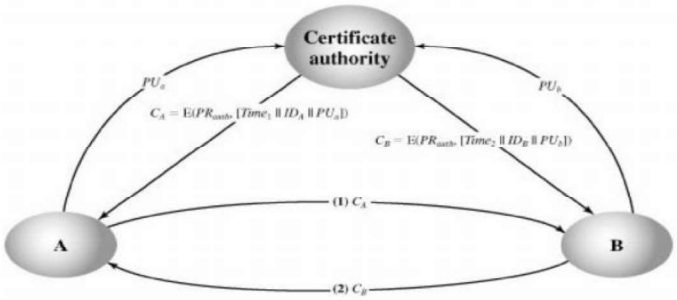
Publicly available directory of Public Keys



Public-key authority of Public Keys



Public-Key Certificates of Public Keys



RSA

Key Generation

1. Generate two large prime numbers, p and q
2. Let n = pq
3. Let m=Φ(n) = (p-1)(q-1)
4. Choose a small number e, co prime to m, with GCD (Φ(n), e)= 1; 1<e<Φ(n)
5. Find d, such that de mod Φ(n) = 1

Publish e and n as the public key.
Keep d and m as the secret key.

Encryption

cipher = (message)^e mod n

Decryption

message= (cipher)^d mod n

x mod y means the remainder of x divided by y

RSA

Select Two Primes P & Q

$P = 53$ $Q = 59$

Start to calculate first part of the Public Key = n

$PQ = n$ Or $53 \times 59 = 3127$

RSA

We also need a small exponent e

e must:

1) Be an Integer
2) Not be a factor of n
3) $1 < e < \varphi(n)$
   For this example $e = 3$

RSA

Select Two Primes P & Q

$P = 53$ $Q = 59$

Start to calculate first part of the Public Key = n

$PQ = n$ Or $53 \times 59 = 3127$

We also need a small exponent e

e must:

1) Be an Integer
2) Not be a factor of n
3) $1 < e < \varphi(n)$
   For this example $e = 3$

Our public key is made up of n & e or 3127 & 3

RSA

We need to calculate $\varphi(n)$

$$\varphi(n) = (P-1)(Q-1)$$

$$\varphi(n) = (53-1)(59-1)$$

$$\varphi(n) = 3016$$

RSA

Now Calculate Private Key = d

$$d = \frac{2(\varphi(n)) + 1}{e}$$

$$2011 = \frac{2(3016) + 1}{3}$$

RSA

We need to calculate $\varphi(n)$

$$\varphi(n) = (P - 1)(Q - 1)$$

$$\varphi(n) = (53 - 1)(59 - 1)$$

$$\varphi(n) = 3016$$

Now Calculate Private Key = d

$$d = \frac{2(\varphi(n)) + 1}{e}$$

$$2011 = \frac{2(3016) + 1}{3}$$

Private Key

$d = 2011$

RSA

We now have everything we need

1) Our public key is made up of n & e or 3127 & 3
2) Our Private Key of d or 2011

A simple example. Lets encrypt "HI"
Convert letters to numbers: H = 8 & I = 9
"89"

c = Encrypted Data

$$c = 89^e \bmod n$$

$$1394 = 89^3 \bmod 3127$$

RSA

Decrypt the data with the Private Key

c = Encrypted Data
d = Private Key
n = Public Key

Decrypted Data = $c^d \bmod n$

$$89 = 1394^{2011} \bmod 3127$$

MODULE4:Cryptographic hash function (CHF)

- A mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit array of a fixed size ("hash value", "hash", "message digest").
- Only way to find a message produced by hash is to attempt a brute-force search
- The ideal cryptographic hash function has the following main properties:
  - It is deterministic, meaning that the same message always results in the same hash
  - It is quick to compute the hash value for any given message
  - It is infeasible to generate a message that yields a given hash value
  - It is infeasible to find two different messages with the same hash value
  - Small change to a message should change the hash value

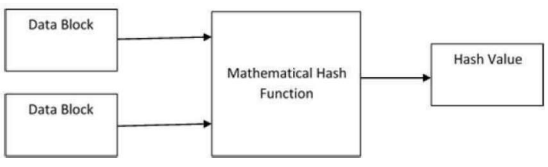
Cryptographic hash function (CHF)

- **Features of Hash Functions**:
- **Fixed Length Output (Hash Value)**
  - Hash function coverts data of arbitrary length - fixed length, referred as **hashing the data**
  - Hash is smaller than the input data, hence sometimes called **compression functions**
  - Hash is a smaller representation of a larger data, it is also referred to as a **digest**.
  - Hash function with n bit output is referred to as an **n-bit hash function**. Popular hash functions generate values between 160 and 512 bits.
- **Efficiency of Operation**
  - Generally for any hash function h with input x, computation of h(x) is a fast operation.
  - Computationally hash functions are much faster than a symmetric encryption.

Properties of Hash Functions

- **Pre-Image Resistance**
  - It should be computationally hard to reverse a hash function.
  - If a hash function h produced a hash value z for input x, then it is difficult process to find
  - This property protects against an attacker who only has a hash value
- **Second Pre-Image Resistance**
  - If a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This protects against an attacker who has an input value and its hash, and wants to substitute different input value.
- **Collision Resistance (Collision free hash function)**
  - It is hard to find two different inputs of any length that result in the same hash.
  - If a hash function is collision-resistant **then it is second pre-image resistant.**

Design of Hashing Algorithms



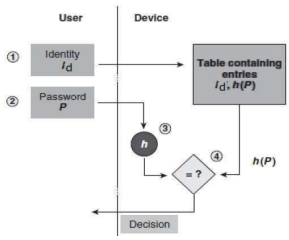
- The size of each data block varies depending on the algorithm (block sizes are from 128 bits to 512 bits.
- Hashing algorithm involves rounds of hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round

Popular Hash Functions

- Message Digest (MD)
- Secure Hash Function (SHA)
- RIPEMD (RACE Integrity Primitives Evaluation Message Digest)
- Whirlpool

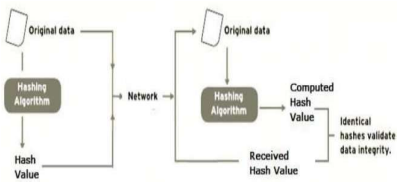
Applications of Hash Functions

- Password Storage:
- Instead of storing password, mostly all logon processes store the hash values of passwords in the file.
- The Password file consists of a table of pairs which are in the form (user id, h(P)).



Applications of Hash Functions

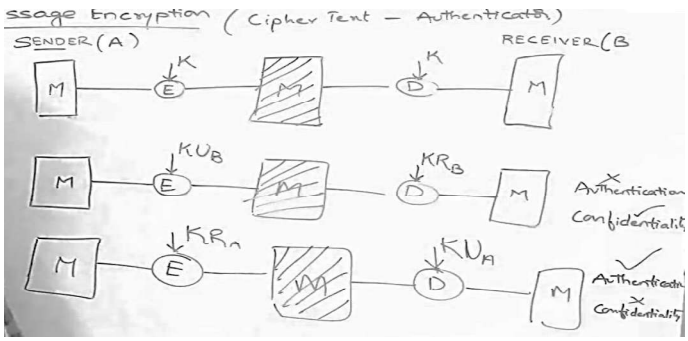
- Data Integrity Check
- It is used to generate the checksums on data files.
- Helps the user to detect any changes made to original file. The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver.



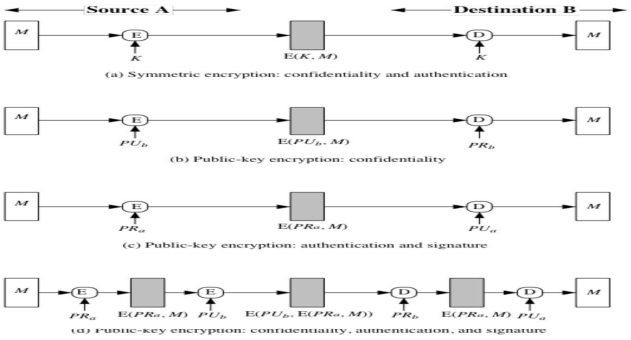
Types of Authentication

- Message authentication
  - Ciphertext
- Message authentication code (MCA)
  - C(M,K)= Fixed length code
  - MAC, Message digest
- Hash function
  - H)M)= Fixed length code(Hash code 'h')

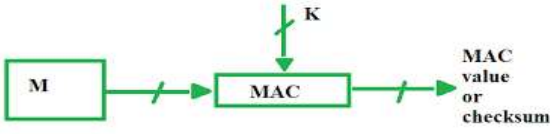
Message Authentication



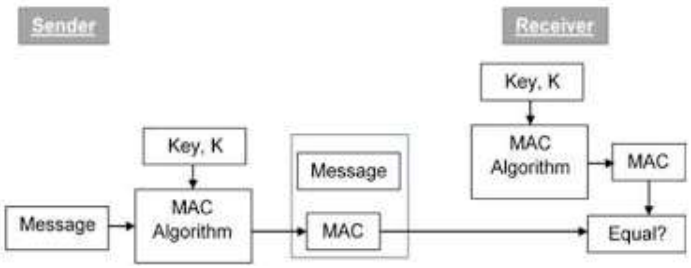
Message Authentication



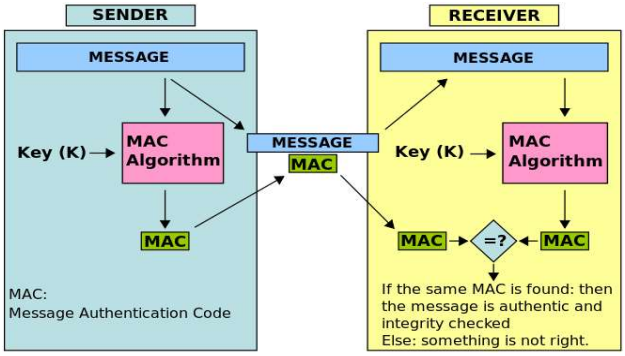
Message Authentication code

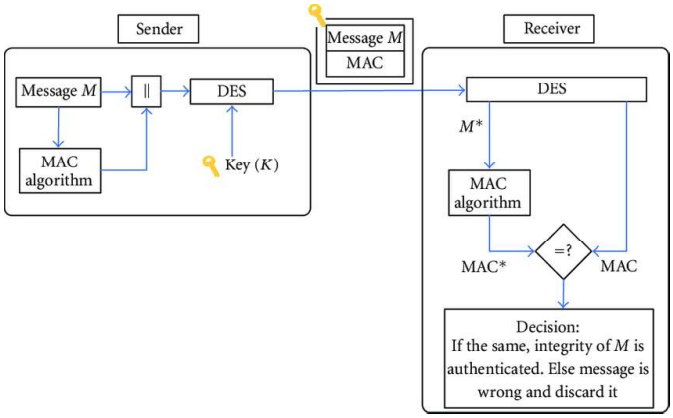


Message Authentication code



Message Authentication code





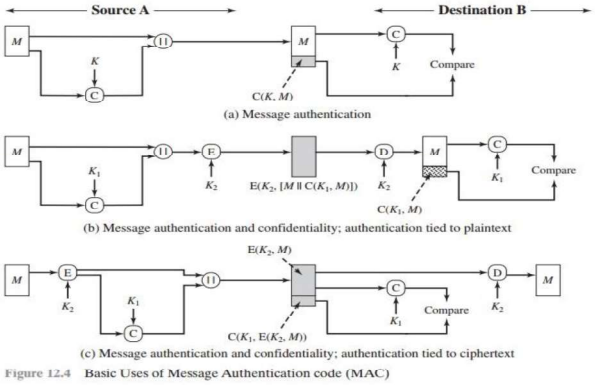
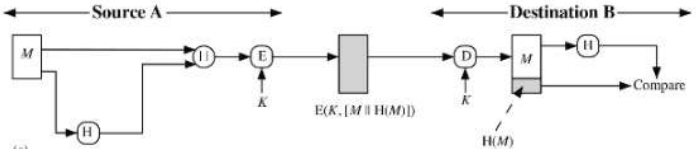
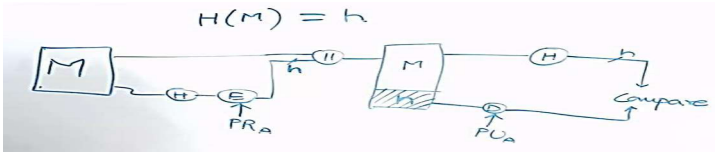


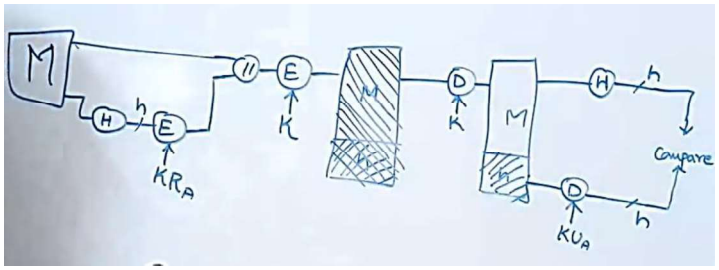
Figure 12.4 Basic Uses of Message Authentication code (MAC)

Hash function(Hash code Authenticator)

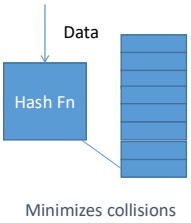




Hash function(Hash code Authenticator)



Hash Function



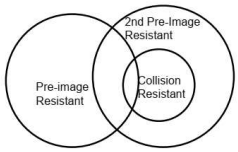
- Hash tables used in data searches
- Purpose:

produce a fixed-size "fingerprint" or digest of arbitrarily long input data .

- Why? To guarantee integrity
- Properties
- Take variable size input, Produce fixed output size (Size of the table),Be easy to compute
- Be pseudorandom so that it distributes uniformly over the table

Non cryptographic Properties of Hash Functions

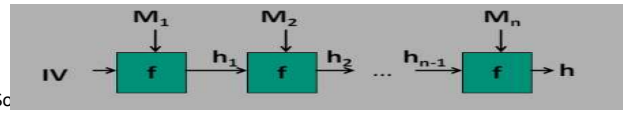
"Good" Hash Function



- Given any h, computationally infeasible to find any x such that H(x) = h → Pre-image Resistant = one-way
- For a given x, computationally infeasible to find y such that H(y) = H(x) and y≠x → 2nd Pre-image Resistant = Weak Collision Resistant
- Computationally infeasible to find any (x, y) such that H(x) = H(y) and x ≠ y → Strong Collision Resistant

Construction

- A hash function is typically based on an internal compression function f() that works on fixed-size input blocks (M_i)



- Sc

- Produces a hash value for each fixed-size block based on its content and hash value for the previous block
- "Avalanche" effect: 1-bit change in input produces "catastrophic" and unpredictable changes in output

Simple Hash Functions

- Bitwise-XOR
- Not secure
- Can be improved by rotating the hash code after each block is XOR-ed into it
- If message itself is not encrypted, it is easy to modify the message and append one block that would set the hash code as needed
- Another weak hash example: IP Header CRC

Simple Hash Functions

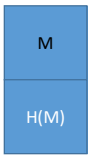
	bit 1	bit 2	• • •	bit *n*
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
block *m*	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

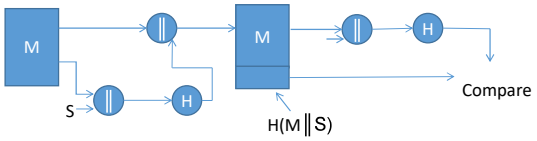
Examples of Hash Functions

- MD4 = Message Digest 4 [RFC 1320] - 32b operations
- MD5 = Message Digest 5 [RFC 1321] - 32b operations
- SHA = Secure hash algorithm [NIST]
- SHA-1 = Updated SHA
- SHA-2 = SHA-224, SHA-256, SHA-384, SHA-512
- SHA-512 uses 64-bit operations

Applications of crypto Hash

- **Message Authentication** = Integrity





Applications of crypto Hash

Digital Signatures

- **Message Authentication** = Integrity
- **Digital Signatures**: Encrypt hash with private key
- **Password storage**: Hash of the user's password is compared with that in the storage. Hackers can not get password from storage.
- **Pseudorandom number generation**: Hash an IV, Hash the hash, …, repeat

Birthday Problem(Cont)

What is the probability that two people have the same birthday (day and month)

- With 30 people in a room, what is chance that two people have a common birthday
- P(s) = probability of at least 2 people have the same birthday)
- P(d) = probability of someone share with at least someone else birthday)
- P(s)+P(d)=1
- P(s)= 1- P(d) = 100% - P(d)
- For 30 people:
  P(d) = 365 x 364 x 356 x…. 336 / (365)!
  = 365! x (365 – n)! / (365)^n

- In general, n possibilities
  ⇒ $\sqrt{n}$ trials to find a collision

Birthday Problem(Cont)

- With 30 people in a room, what is chance that two people have a common birthday
- P(s) = probability of at least 2 people have the same birthday)
- P(d) = probability of someone share with at least someone else birthday)
- P(s)+P(d)=1
- P(s)= 1- P(d) = 100% - P(d)
- For 30 people:
  P(d) = 365 x 364 x 356 x…. 336 / (365)!
  = 365! x (365 – n)! / (365)^n
  = 365! x (365 – 30)! / (365)^{30}
- P(s)= 1- P(d) = 100% - P(d) = 100- 29.37= 70.63%

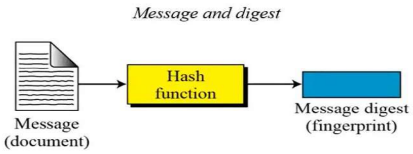
- In general, n possibilities
  ⇒ $\sqrt{n}$ trials to find a collision

MD5

> Integrity of the message means no modification of any kind in the message. For preserving the integrity of any message concept of fingerprint can be used and attached with the message.

MD5

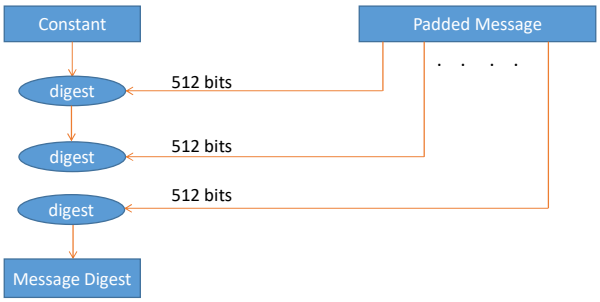
The electronic equivalent of the document and fingerprint pair is the message and digest pair.



Message Digest version 5(MD5)

- Author: R. Rivest, 1992
- 128-bit hash
  - based on earlier, weaker MD4 (1990)
- **Collision resistance (Birthday attack resistance)**
  - only 64-bit
- Output size not long enough today (due to various attacks)
- Overview:
  1, MD
  2. Append MD to plain text
  3. Send to receiver
  4.MD – Receiver
  5. Compare 1 and 4

Overview of MD5



MD5

- PT 512
- Append padding bit to make 512bits
- Append 64 bit representation, then PT length should be multiples of 512 bits
- Eg. PT is 512, then take 448 then append 64 bits
- If less than 448, Eg. 440 then pad 8 bits, then add 64 bits
- Initialize MD buffers (Size 32 bits, 4 buffers- A, B,C, D) 128 bits (o/p of MD5)
- Process each 512 bit block
- Output – message digest in buffers

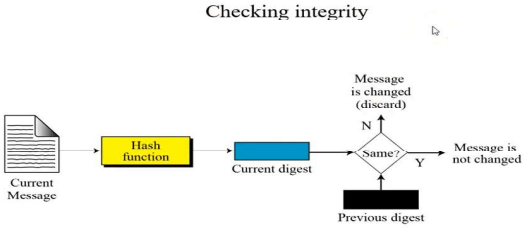
MD5

The two pairs (document / fingerprint) and (message / message digest) are similar, with some differences. The document and fingerprint are physically linked together. The message and message digest can be unlinked separately, and, most importantly, the message digest needs to be safe from change.

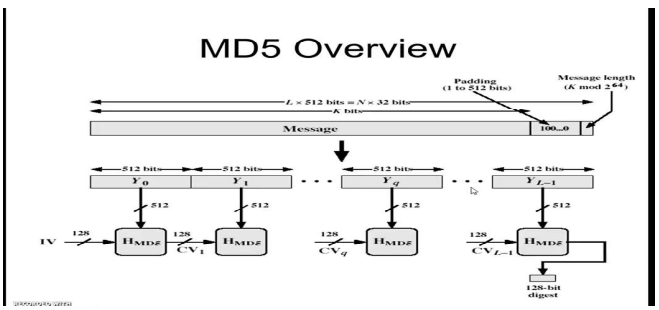
Example:

MD5("The quick brown fox jumps over the lazy dog") = *9e107d9d372bb6826bd81d3542a419d6*
MD5("The quick brown fox jumps over the lazy cog") = *1055d3e698d289f2af8663725127bd4b*

MD5



MD5



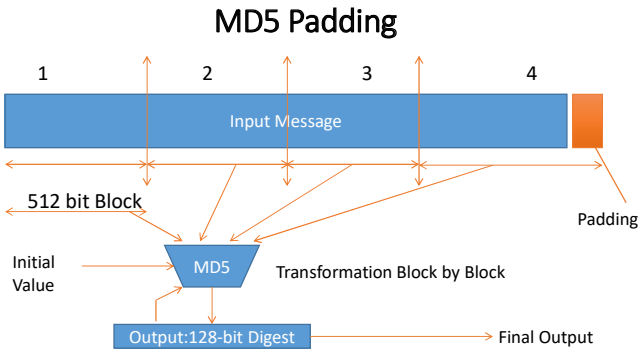
MD5 Padding

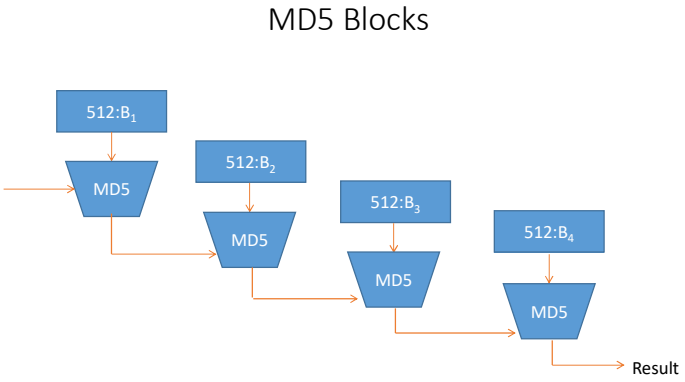
- Given original message M, add padding bits "100…" such that resulting length is 64 bits less than a multiple of 512 bits.
- Append *original length in bits* to the padded message
- Final message chopped into 512-bit blocks

1-512 bits 64 bits

Original Message	1000…000	Original length in bits

Multiple of 512 bits





MD5

Implementation Steps

- Step3. Initialize MD buffer
  A four-word buffer (A, B, C, D) is used to compute the message digest. Each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

  word A: 01 23 45 67
  word B: 89 ab cd ef
  word C: fe dc ba 98
  word D: 76 54 32 10

Buffer value

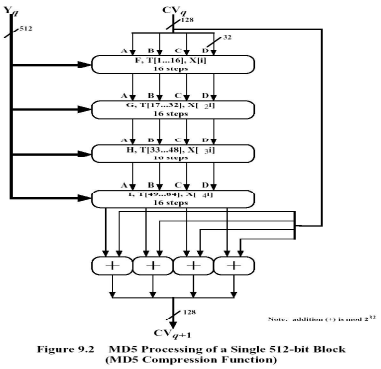


Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

MD5

Implementation Steps

- Step4. Process message in 16-word blocks
  Four functions will be defined such that each function takes an input of three 32-bit words and produces a 32-bit word output.
  - F(x,y,z) == (x∧y)∨(~x ∧ z) (Round1)
  - F(x,y,z) == (x ∧ z) ∨(y ∧~ z) (Round2)
  - F(x,y,z) == x⊕y⊕ z (Round3)
  - F(x,y,z) == y⊕(x ∧ ~z) (Round4)

MD5 Compression Function

- Each round has 16 steps of the form:
  `a = b+((a+F(b,c,d)+M[i]+T[k])<<<s)`
- a,b,c,d refer to the 4 words of the buffer, but used in varying permutations
- where F(b,c,d) is a different nonlinear function in each round (1,2,3,4)

MD5 Circular left shift

[ABCD]
[DABC]
[CDAB]
[BCDA]
[ABCD]
[DABC]
[CDAB]
[BCDA]
[ABCD]
[DABC]
[CDAB]
[BCDA]
[ABCD]
[DABC]
[CDAB]
BCDA

Buffer value

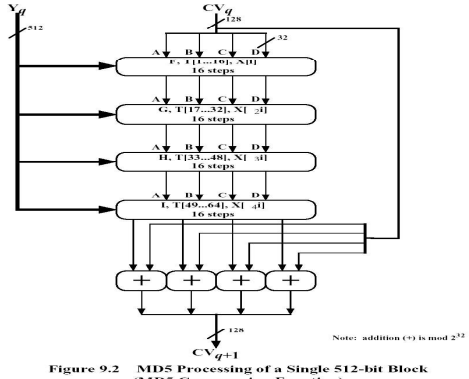


Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

MD5 compression function

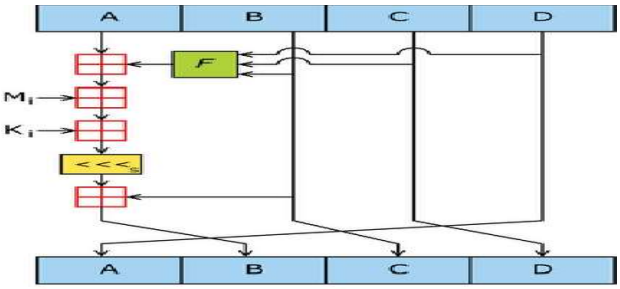
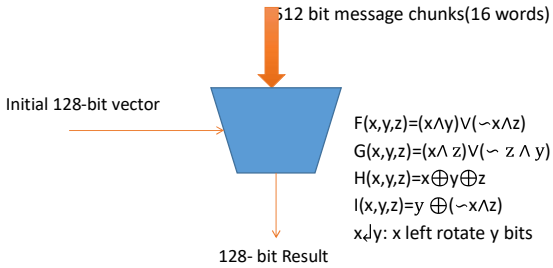


Fig. 1. Typical MD5

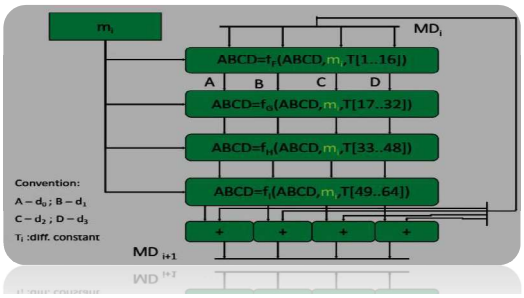
MD5 Box



MD5 Process

- As many stages as the number of 512-bit blocks in the final padded message
- Digest: 4 32-bit words: MD=A|B|C|D
- Every message block contains 16 32-bit words: m0|m1|m2 …|m15
  - Digest MD_0 initialized to: A=01234567,B=89abcdef,C=fedcba98, D=76543210
  - Every stage consists of 4 passes over the message block, each modifying MD; each pass involves different operation

Processing of Block m_i



Secure Hash Algorithm(SHA)

- Successor to and similar to MD5 (by Ron Rivest)
- SHA-0: FIPS PUB 180, 1993. Withdrawn shortly after publ.
- SHA-1: FIPS PUB 180-1, 1995. 160 bit hash
- SHA-2: FIPS PUB 180-2, 2002
  - SHA-224 ,SHA-256,SHA-384,SHA-512
- SHA-1 is used in TLS, SSL, PGP, SSH, S/MIME, and IPsec
  - Required by law in US Govt applications
  - Used in Digital Signature Standard
- Pseudo-codes for SHA algorithms are available.
- NIST certifies implementations.

SHA-Algorithm

- Fixed length value has been generated which provides the authentication and it is independent of key
- Hash algorithm- Secure hash algorithm
  - If output = 128 bits – SHA1, If output =256 bits – SHA 256, and If output = 512 bits – SHA 512
  - H(M)= h(512 bits)

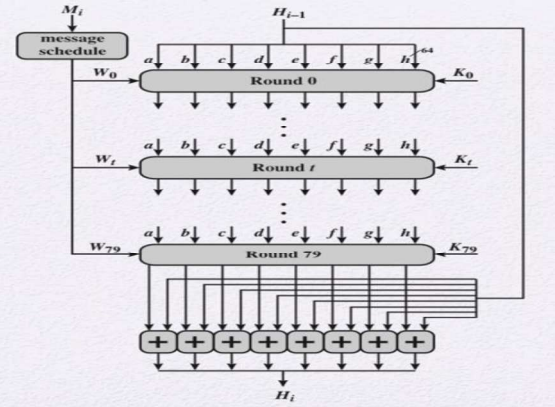
SHA-512 Algorithm

- PT block size is 1024 bits
- No. of rounds = 80
- Each round = Qword = 64 bits(Q word is generated from PT)
- Each round uses K constants, it use 8 buffers to store the intermediate result
  - Output is the hash code
- Each buffer – 64 bits (total 64 x 8= 512 bits)

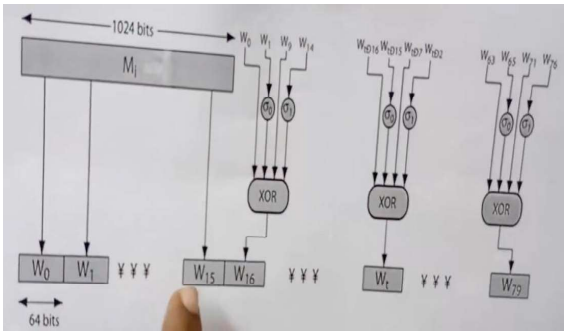
SHA-512 Algorithm - steps

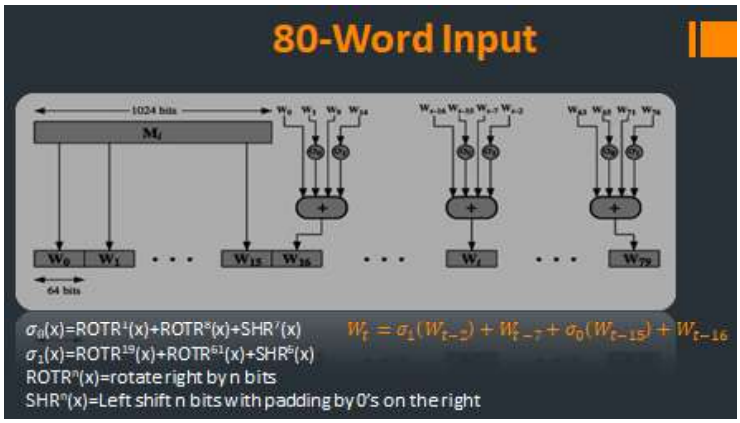
- PT block size is 1024 bits, Padding should be done if needed (1000…), ie. PT should be 128 bits < multiples of 1024 bits
- Append 128 bit representation, then PT length should be multiples of 1024 bits
- Initialize buffers (a, b, c, d, e, f, g, h) , Size 64 bits hexa decimal, 8 buffers 512 bits (o/p)
- Process each block of PT in 80 rounds
- Output – hash code (512 bits)

SHA- 512 Round Function



SHA- 512 Qword





Key constants

- 80 constants (20 x 4 = 80)
- Apply random function to select key for every round of iteration

SHA-512 Round constants (K)

428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

Figure: List of round constants used in SHA-512

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

Majority Function

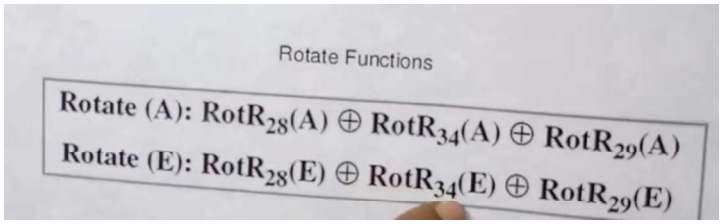
$$(A_j \text{ AND } B_j) \oplus (B_j \text{ AND } C_j) \oplus (C_j \text{ AND } A_j)$$

Conditional Function

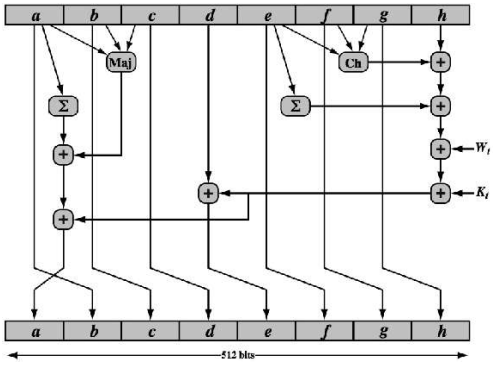
$$(E_j \text{ AND } F_j) \oplus (\text{NOT } E_j \text{ AND } G_j)$$

Rotate Functions

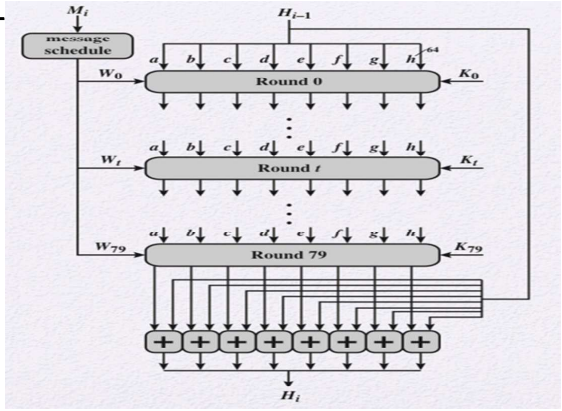
$$\text{Rotate (A): } \mathrm{RotR}_{28}(A) \oplus \mathrm{RotR}_{34}(A) \oplus \mathrm{RotR}_{29}(A)$$
$$\text{Rotate (E): } \mathrm{RotR}_{28}(E) \oplus \mathrm{RotR}_{34}(E) \oplus \mathrm{RotR}_{29}(E)$$



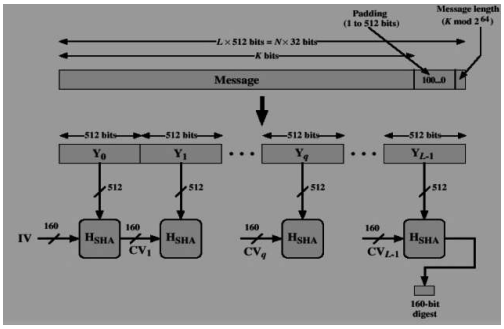
SHA- 512



SHA-



Digest generation with SHA- 512



General Logic

- Input message must be < 264 bits
  - not a real limitation
- Message processed in 512-bit blocks sequentially
- Message digest (hash) is 160 bits
- SHA design is similar to MD5, but a lot stronger
- Basic Steps
  Step1: Padding
  Step2: Appending length as 64-bit unsigned
  Step3: Initialize MD buffer: 5 32-bit words: A|B|C|D|E
  A = 67452301 B = efcdab89 C = 98badcfe D = 10325476
  E = c3d2e1f0

- 0 <= t <= 19
  K_t = 5A827999
- 20<=t<=39
  K_t = 6ED9EBA1
- 40<=t<=59
  K_t = 8F1BBCDC
- 60<=t<=79
  K_t = CA62C1D6

Basic Steps Cont.,

- Step 4: the 80-step processing of 512-bit blocks: 4 rounds, 20 steps each
- Each step t (0 <= t <= 79):
  - Input:
    - W_t – 32-bit word from the message
    - K_t – constant
    - ABCDE: current MD
- Output:
  - ABCDE: new MD
- Only 4 per-round distinctive additive constants:

Basic Logic Functions

- Only 3 different functions
- Round — Functions $f_t(B,C,D)$

Round	Functions $f_t(B,C,D)$
0 <= t <= 19	(B∧C)∨(∼B∧D)
20<=t<=39	B⊕C⊕D
40<=t<=59	(B∧C)∨(B∧D) ∨(C∧D)
60<=t<=79	B⊕C⊕D

- Additional mixing used with input message 512-bit block
  - $W_0|W_1|\ldots|W_{15} = m_0|m_1|m_2\ldots|m_{15}$
  - For 15 < t <80:
  - $W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$

SHA-1 Versus MD5

- SHA-1 is a stronger algorithm:
  - A birthday attack requires on the order of 2^{80} operations, in contrast to 2^{64} for MD5
- SHA-1 has 80 steps and yields a 160-bit hash (vs. 128) - involves more computation

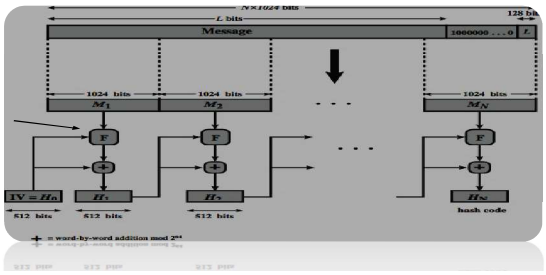
SHA-2 Algorithm

- SHA-256 uses 32-bit operations
- SHA-512 uses 64-bit operations
- Use different shift amounts and additive constants
- SHA-224 and SHA-384 are simply truncated versions of SHA- 256 and SHA-512 using different initial values.
- SHA-224 matches the key length of two-key triple-DES

SHA- 512

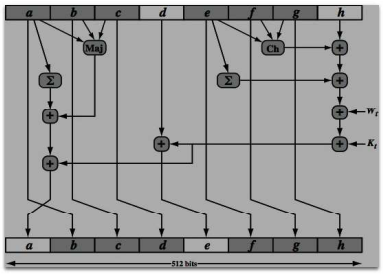
Append Padding bits , Append Length

80 Rounds



SHA 512 Round Function

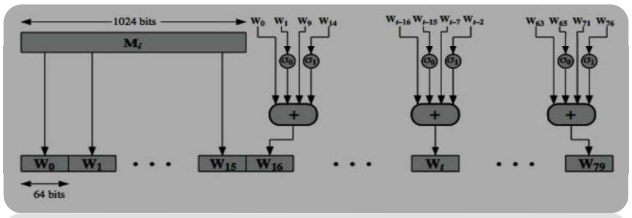
Conditional Fn Ch(e,f,g):
if e then f else g
= (e AND f) ⊕ (Not e and g)
Majority Fn Maj(a, b, c): True if 2 of 3 args are true
=(a AND b) ⊕(a AND c) ⊕(b AND c)



Input Sequence word

Constant

80-Word Input



σ_0(x)=ROTR^1(x)+ROTR^8(x)+SHR^7(x)
σ_1(x)=ROTR^{19}(x)+ROTR^{61}(x)+SHR^6(x)
ROTR^n(x)=rotate right by n bits
SHR^n(x)=Left shift n bits with padding by 0's on the right

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

SHA-3

- SHA-2 (esp. SHA-512) seems secure
  - Shares same structure and mathematical operations as predecessors so have concern
- NIST announced in 2007 a competition for the SHA-3
  - Has had 3 rounds of narrowing down the selections
  - Five algorithms advanced to the third (and final) round in December 2010
  - Final selection to be announced by 2012

Practice

- Compute the following hash function:

  $h = (7 + \sum_{i=1}^{k}(m_i)^2) \bmod 251$

  for a 4-byte message M={m1, m2, m3, m4}={128, 252, 33, 19}
  All are decimal numbers.
- Check if the hash function is:
  A. Collision Resistant
  B. Pre-image resistant
  C. Second Pre-image Resistant
- Show counter examples for any property that is not satisfied.

Message Authentication

- Use symmetric encryption such as AES or 3-DES
  - Generate H(M) of same size as E() block
  - Use E_K(H(M)) as the MAC (instead of, say, DES MAC)
  - Alice sends E_K(H(M)) , M
  - Bob receives C,M' decrypts C with k, hashes result
    $H(D_K(C)) =?= H(M')$

Collision ⟶ MAC forgery!

MAC Requirements

MAC Requirements

1. Large enough entropy
2. Collision resistance
3. MAC(K,M) is uniformly distributed
4. Avalanche Effect

HMAC

Hash for Authentication

- Alice and Bob share a secret key K_{AB}
  1. Alice → Bob: random challenge r_A
  2. Bob → Alice: $H(K_{AB} || r_A)$, random challenge r_B
  3. Alice → Bob: $H(K_{AB} || r_B)$

  Only need to compare H() results
- Cannot just compute and append H(m)
- Need "Keyed Hash":
  - Prefix:
    - MAC: H(KAB | m), almost works, but …
    - Allows concatenation with arbitrary message, H(KAB | m | m')

Using Hash to compute MAC

HMAC

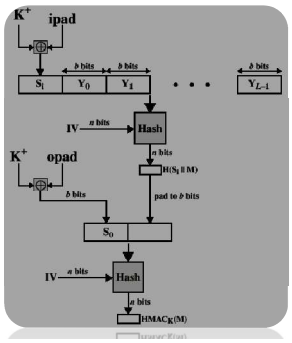
Main Idea:

- Use a MAC derived from any cryptographic hash function
- hash functions do not use a key, therefore cannot be used directly as a MAC

Motivations for HMAC:

- Cryptographic hash functions execute faster in software than encryption algorithms such as DES
- No need for the reverse ability of encryption

HMAC Algorithm

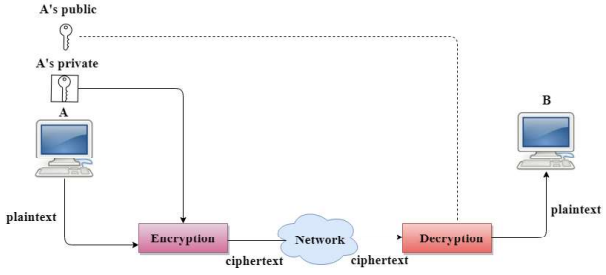


- Compute H1 = H() of the concatenation of M and K1
- To prevent an "additional block" attack, compute again H2= H() of the concatenation of H1 and K2
- K1 and K2 each use half the bits of K
- Notation:
  - K^+ = K padded with 0's
  - ipad=00110110 x b/8
  - opad=01011100 x b/8
  - Execution: Same as H(M), plus 2 blocks

MODULE4:digital signature

- The **Digital Signature** is a technique which is used to validate the authenticity and integrity of the message. Four aspects of **security**:
  - privacy, authentication, integrity, and non-repudiation.
  - The basic idea behind the **Digital Signature** is to sign a document.
  - RSA and DSS or DSA approaches are used

- **RSA** is a commonly used scheme for **digital signatures**. **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**
- **Digital Signature Algorithm (DSA)** is one of the Federal Information Processing Standard for making digital signatures depends on the mathematical concept or we can say the formulas of modular exponentiation and the discrete logarithm problem to cryptograph the signature digitally in this algorithm.



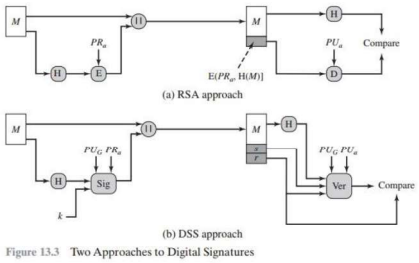
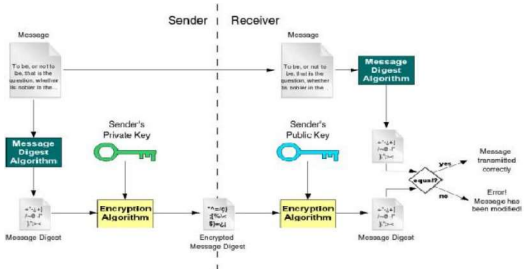
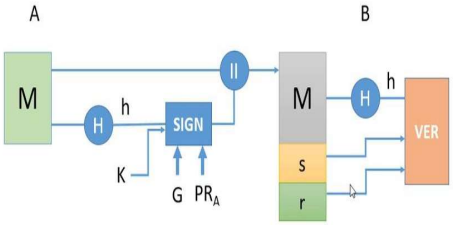


Figure 13.3 Two Approaches to Digital Signatures

RSA



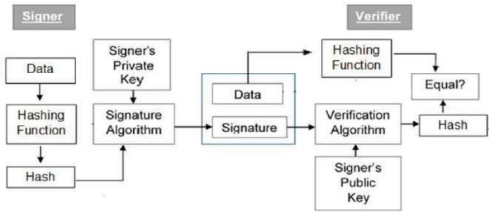
Digital Signature – DSS Approach

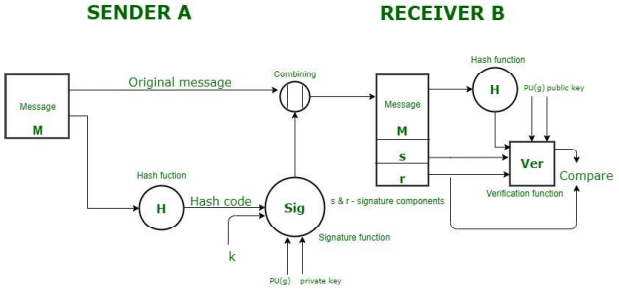


Signature = {s, r}

K → Random number

G → Global element





Global Public-Key Components

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

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature = (r, s)

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

Digital Signature Algorithm

Global Public-Key Components

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

q prime divisor of $(p-1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(\mathrm{H(M)} + xr)] \bmod q$

Signature = (r, s)

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [\mathrm{H(M')}w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M = message to be signed

$\mathrm{H}(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

Module5:Symmetric key distribution using symmetric encryption

- For symmetric encryption to work, the two parties must share the same key, and that key must be protected from access by others
- Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key
- Therefore, the strength of any cryptographic system rests with the ***key distribution technique*** - a term refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key

Symmetric key distribution using symmetric encryption

- Key distribution between two parties A & B can be achieved as follows:
  1. A can select a key and physically deliver it to B
  2. A third party can select the key and physically deliver it to A and B
  3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key
  4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B

…Symmetric key distribution using symmetric encryption

- Option 1 and 2 call for manual delivery of a key
  - For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link
  - Not suitable for end-to-end encryption over network
  - In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time
  - Each device needs a number of keys supplied dynamically
  - If there are N hosts, the number of required keys are ***[N(N - 1)]/2***

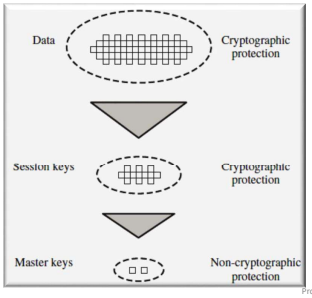
…Symmetric key distribution using symmetric encryption

- Option 3 is a possibility for either link encryption or end-to-end encryption
- If an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed
- the initial distribution of potentially millions of keys still must be made

…Symmetric key distribution using symmetric encryption

- In option 4 scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed
- Each user must share a unique key with the key distribution center for purposes of key distribution
- The use of a key distribution center is based on the use of a hierarchy of keys
- At a minimum, two levels of keys are used : session key and master key

…Symmetric key distribution using symmetric encryption



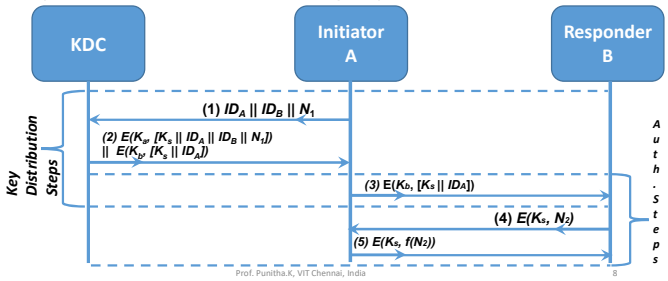
- Communication between end systems is encrypted using a temporary key, often referred to as a session key
- Session keys are transmitted in encrypted form, using a master key that is shared by the key distribution center and an end system or user
- For each end system or user, there is a unique master key that it shares with the key distribution center

…Symmetric key distribution using symmetric encryption

- **A Key distribution scenario**
  - Assumes that each user shares a unique master key with the key distribution center (KDC)
  - user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection
  - A has a master key, $\mathbf{K_a}$, known only to itself and the KDC; similarly, B shares the master key $\mathbf{K_b}$ with the KDC

…Symmetric key distribution using symmetric encryption

- **…A Key distribution scenario : following steps occur**



…Symmetric key distribution using symmetric encryption

- **Hierarchical Key Control**
  - It is not necessary to limit the key distribution function to a single KDC
  - For very large networks, a hierarchy of KDCs can be established
  - E.g. local KDCs, responsible for key distribution for a small domain of the overall internetwork, such as a single LAN or a single building

…Symmetric key distribution using symmetric encryption

- **…Hierarchical Key Control**
  - If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC
  - minimizes the effort involved in master key distribution, because most master keys are shared by a local KDC with its local entities

…Symmetric key distribution using symmetric encryption

- **Session Key Lifetime**
  - The more frequently session keys are exchanged, the more secure they are - opponent has less ciphertext to work with for any given session key
  - the distribution of session keys delays the start of any exchange and places a burden on network capacity
  - A security manager must try to balance the competing considerations

…Symmetric key distribution using symmetric encryption

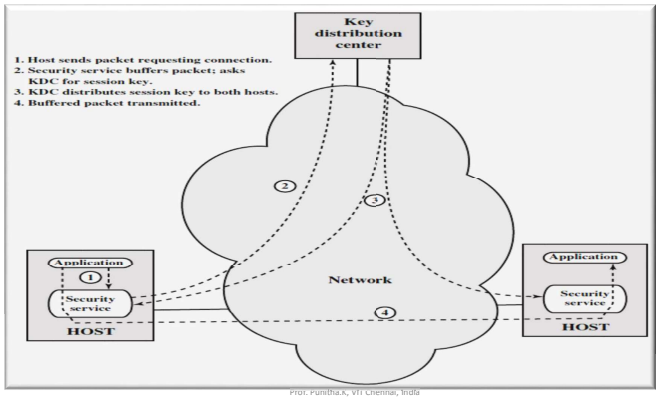
- **…Session Key Lifetime**
  - For connection-oriented protocols, use a new session key for each new session
  - For connection less protocol (i.e. transaction-oriented protocol), the most secure approach is to use a new session key for each exchange
  - A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions

…Symmetric key distribution using symmetric encryption

- **A Transparent Key Control Scheme**
  - The scheme is useful for providing end-to-end encryption at a network or transport level

The steps involved in establishing a connection are

1. Host sends packet requesting connection
2. Session Security Module (SSM) buffers packets; asks KDC for session key
3. KDC distributes session key to both hosts
4. Buffered packets transmitted

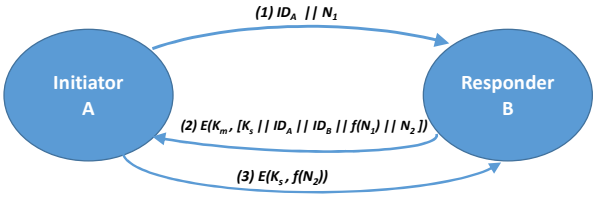


…Symmetric key distribution using symmetric encryption

- **Decentralized Key Control**
  - Key distribution center imposes the requirement that the KDC be trusted and be protected from subversion
  - This can be avoided if key distribution is fully decentralized
  - Each end system be able to communicate in a secure manner with all potential partner end systems
  - *[n(n - 1)]/2* master keys for a configuration with 'n' end systems

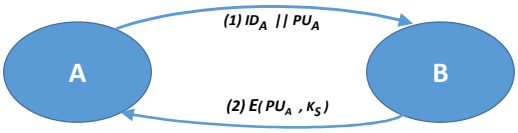
…Symmetric key distribution using symmetric encryption

- **…Decentralized Key Control :** A session key may be established with the following sequence of steps



Symmetric key distribution using Asymmetric encryption

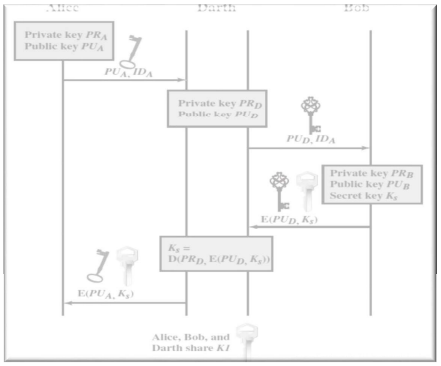
- One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution
- ***Simple Secret Key Distribution***
  - If A wishes to communicate with B, the following figure depicts the same



…Symmetric key distribution using Asymmetric encryption

- ***…Simple Secret Key Distribution***
  - The risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping
  - ***Man-in-the-Middle Attack***: Insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message
  - An adversary, D, has control of the intervening communication channel, then D can compromise the communication without being detected

Man-in-the-Middle Attack



The result is that both Alice and Bob know K_S and are unaware that K_S has also been revealed to Darth

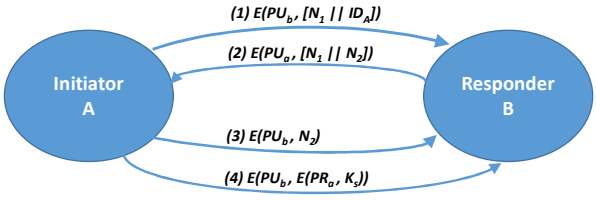
Only threat is eavesdropping

…Symmetric key distribution using Asymmetric encryption

- ***Secret Key Distribution with Confidentiality and Authentication***
  - Provides protection against both active and passive attacks
  - It is assumed that A and B have exchanged public keys
  - This scheme ensures both confidentiality and authentication in the exchange of a secret key
  - Next slide shows the steps involved in this scheme

…Symmetric key distribution using Asymmetric encryption

- ***…Secret Key Distribution with Confidentiality and Authentication***

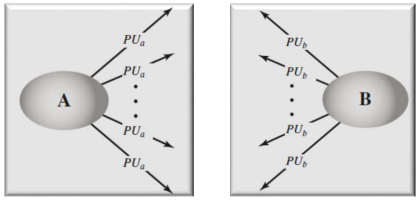


Distribution of Public Keys

- **General Schemes**
  - ***Public announcement***
  - ***Publicly available directory***
  - ***Public-key authority***
  - ***Public-key certificates***

…Distribution of Public Keys

- **Public announcement of Public keys**
  - This approach is convenient - Major weakness: Anyone can forge such a public announcement



…Distribution of Public Keys

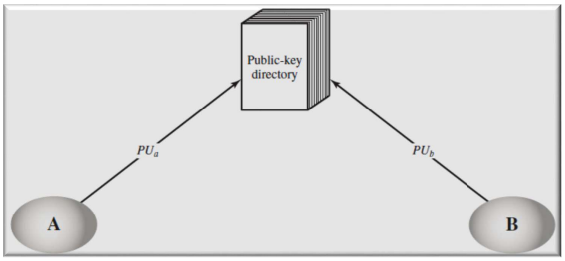
- **Publicly Available Directory**
  - A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys
  - Maintenance and distribution of the public directory :be the responsibility of some trusted entity or organization
  - ***Elements :***
    - The authority maintains a directory with a {name, public key} entry for each participant
    - Each participant registers a public key with the directory authority

…Distribution of Public Keys

- ***Elements :***
  - A participant may replace the existing key with a new one at any time
  - Participants could also access the directory electronically
- ***Vulnerabilities***: an adversary succeeds in obtaining or computing the private key of the directory authority ; adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant

…Distribution of Public Keys

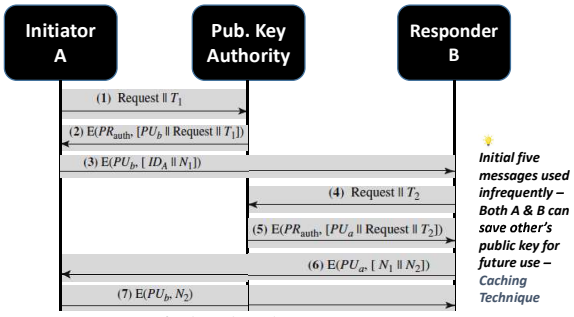
- **…Publicly Available Directory**



…Distribution of Public Keys

- **Public-Key Authority**
  - Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory
  - each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key
  - Next slide shows the steps involved in this scheme

…Distribution of Public Keys



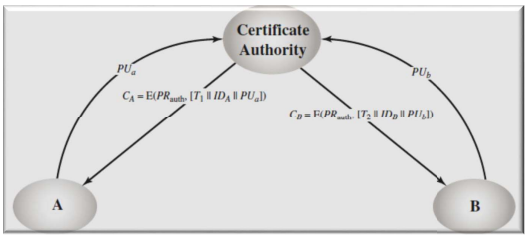
Initial five messages used infrequently – Both A & B can save other's public key for future use – Caching Technique

…Distribution of Public Keys

- **Public-Key Certificates**
  - Previous scheme has some drawbacks: The Public-key authority could be a bottleneck in the scheme; user must appeal to the authority for a public key and it is vulnerable to tampering
  - An alternative approach : use certificates – can be used by participants to exchange keys without contacting a public-key authority
  - A certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party

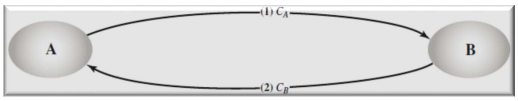
…Distribution of Public Keys

- **Obtaining Certificates from Certificate Authority (CA)**



…Distribution of Public Keys

- **Exchanging Certificates**



- *The timestamp serves as an expiration date*
- *If a certificate is sufficiently old, it is assumed to be expired*
- *One scheme has become universally accepted for formatting public-key certificates: the X.509 standard*

Public-Key Infrastructure (PKI)

- RFC 4949 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography
- The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys
- Interrelationship among the key elements of the PKIX model
  - End entity
  - Certification Authority (CA)

…Public-Key Infrastructure

- Interrelationship among the key elements of the PKIX model
  - Registration Authority (RA)
  - CRL issuer
  - Repository
- PKIX Management Functions
  - Registration
  - Initialization
  - Certification
  - Key pair recovery

…Public-Key Infrastructure

- PKIX Management Functions
  - Key pair update
  - Revocation request
  - Cross certification

Module6:Visualization of social networks

- Visualization is the graphical display of data with the aim to give insight in that data
- Visualization relies on the extremely powerful human visual system that enables us to see patterns, detect outliers, identify gaps, and make comparisons in the blink of an eye.
- Visualization is not limited to creating static images:
  - a visualization becomes especially useful when it is interactive;
  - allowing users to filter, zoom, correlate, aggregate, and compare the data, i.e. when it allows the user to "play with the data".

Visualization of social networks

- As the fast development of the Internet, many online social network services are created to connect social relationships among people.
- Versatile visualization skills are employed to facilitate analyzing online social networks.
- When the attributes of sociality are concerned, online social networks can be classified into different social communities. In addition, from the daily social activities of various online social communities, social networks may evolve into different patterns of social structure.
- Therefore, visualizations of online social networks were developed according to the attributes of network sociality to present their network structure.

Visualization of social networks

- Analyze the visualization of online social networks according to their attributes of sociality, including:
- Web communities
- Email groups
- Digital libraries and
- Web 2.0 services.

Visualization of social networks

- Web communities:
- Many social network websites are developed with interactive visualization interfaces to facilitate people connecting their communities and maintaining social relationships.
- The Club Nexus was established in 2003, community thus provided very rich profiles for its users to allow for detailed social network analysis, including identifying activities and preferences that determine the formation of friendship.
- In addition to listing actors with their profiles for social network analysis, a modern visualization of social networks, Vizster, was contributed with customized techniques to visualize social relationships and community structures in 2005.

Visualization of social networks

- Web communities:
- Vizster was developed based on node-edge network layouts for exploring connectivity in large graph structures, supporting visual search and analysis, and automatically identifying and visualizing community structures.
- Form the visualization design of Vizster, many advanced search and interactive navigation techniques are developed on Vizster to facilitate the analysis of social networks, such as highlighting, panning, zooming, and distortion techniques.
- From the previous visualization of Web communities, the visualization techniques are mainly introduced to deal with the complex social relationships based on human-centric or user-centric views.

Visualization of social networks

- Web communities:
- As the development of Semantic Web, a project called FOAF (Friend-of-a-friend) was proposed to visualize such human-centric social relationships based on Semantic Web social metadata.
- With XML/RDF format, the FOAF relations can be explicitly defined for further social network analysis and visualization.
- Recently, Microsoft Research Asia proposed a novel object-level search service, called Entity Cube, to help people discover real-world entities, such as people, locations, and organizations, and explore their social relationships.

Visualization of social networks

- **Email groups**:
- Email service is one of the most popular applications that people often use to connect each other and deliver messages in their daily lives.
- Personal online social networks are thus constructed through people's daily social interactions. For analyzing the social structures of the daily email activities, visualization techniques are employed to explore different patterns.
- In 2004, Soylent was developed to study the social patterns and the temporal rhythms of daily email activities. Through the Soylent visualization, mutual interactions between different users and groups, and their everyday collaboration activities can be clearly displayed.

Visualization of social networks

- **Email groups**:
- Some recurrent patterns discovered in the social networks, such as the onion pattern, the nexus pattern, and the butterfly pattern, can thus suggest regular ways of understanding their interactions.
- In addition to the network view of email visualization, two visual metaphors, Social Network Fragments (SNF) and Post History, were employed to visualize the major two dimensions of email activities: people and time.

Visualization of social networks

- **Digital libraries**:
- Social networks can be mainly analyzed from two aspects: authors and writings.
- **Co-Authorship Networks**
- **Co-Citation Relations**

Visualization of social networks

- **Web 2.0 services** :
- the concept of Web 2.0 was proposed in 2004, online social activities are becoming more prosperous than before.
- Many Web 2.0 applications are popularly accessed by users to connect their social networks, such as Twitter and Facebook.

Visualization of social networks

- **Visualization of online social networks classification:**
- Visualization of online social networks can be further categorized into three types by their social relationships:
  - User-centric visualization
  - Content centric visualization and
  - Hybrid visualization

Applications of social network analysis

Members of the committees of the UK Social Network Association

Dimitris Christopoulos	**Politics**	Relational attributes of political entrepreneurs: a network perspective
Bruce Cronin	**Business**	Director networks and UK firm performance
Martin Everett	**Sociology**	The human factor: Re-organisations in public health policy
Tom Alcott	**(Practitioner)**	
Riccardo De Vita	**International business and economics**	Managing resources for corporate entrepreneurship: the case of Naturis
Bernie Hogan	**Communications**	Collecting social network data to study social activity-travel behaviour: an egocentered approach
Paola Tubaro	**Economic sociology**	Norms, status and the dynamics of advice networks: a case study
Federico Varese	**Criminology**	Mafias on the Move: How Organized Crime Conquers New Territories
Pietro Panzarasa	**Business and management**	Community structure and patterns of scientific collaboration in Business and Management
Elisa Bellotti	**Sociology**	What are friends for? Elective communities of single people

Practical uses of SNA

- Power (how can companies utilise privileged positions)
- Influence (which individuals can control decision-making)
- Isolation (who is excluded from certain situations)
- Knowledge transfer (who can best receive and send information)
- Efficiency (do resources flow through a network well)
- Variance (does network position affect social position)

Practical uses of SNA

- Health
- Sports
- Financial security
- Trade
- Weather
- Research

R tool for social network analysis

- Social Network Analysis is a set of methods used to visualize networks, describe specific characteristics of overall network structure, and build mathematical and statistical models of network structures and dynamics.

R tool for social network analysis

Requirements

- The igraph package provides tools for network analysis. The main goals of the igraph library are to provide a set of data types and functions for
  - pain-free implementation of graph algorithms
  - fast handling of large graphs, with millions of vertices and edges
  - allowing rapid prototyping via high level languages like R.
- The readr package provides a fast and friendly way to read rectangular data (like 'csv', 'tsv', and 'fwf'). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes.
- library(igraph) library(readr)
- The .csv files required.

R tool for social network analysis

Requirements

- Building the Network
- Plotting the Network
- Describing the Network
  - Degree Centrality
  - Closeness Centrality
  - Betweenness Centrality

MODULE:6 PROGRAM SECURITY

- Programming errors with security implications
  - buffer overflows, incomplete access control
- Malicious code
  - viruses, worms, Trojan horses
- Program development controls against malicious code and vulnerabilities
  - software engineering principles and practices
- Controls to protect against program flaws in execution
  - operating system support and administrative controls

- Security
- From the point of view of a program/programmer, how can we look at a software component or code fragment and assess its security?
  - (i.e.) assessing software quality in general.
- Who to analyze the software?
  - One person may decide that code is secure because it takes too long to break through its security controls.
  - Someone else may decide code is secure if it has run for a period of time with no apparent failures, but a third person may decide that any potential fault in meeting security requirements makes code insecure.

SECURE PROGRAMS

- Security implies some degree of trust that the program enforces expected confidentiality, integrity, and availability.
- Why is it so hard to write secure programs?

Programs have bugs.

Security-relevant programs have security bugs

IEEE TERMINOLOGY FOR QUALITY: BUG, ERROR, FAULTS, AND FAILURES

- A **bug** can be a mistake in interpreting a requirement, a syntax error in a piece of code, or the cause of a system crash.
- When a human makes mistake, called an **error**, in performing some software activity, the error may lead to a **fault**, or an incorrect step, command, process, or data definition in a computer program.
- A **failure** is a departure from the system's required behavior. A fault is an inside view of the system, as seen by the eyes of the developers, whereas a failure is an outside view: a problem that the user sees.

FINDING AND FIXING FAULTS

- Once you find some faults, fix them
  - Usually by making small edits (called patches) to the program
  - This is called "**penetrate and patch**"
- Patching sometimes makes things worse!
  - Pressure to patch a fault is often high
  - Fault may have caused other failures and a partial fix may cause inconsistencies or other problems
  - The patch for this fault may introduce new faults, here or elsewhere!

- Patch efforts were largely useless, making the system less secure rather than more secure because they frequently introduced new faults.
- There are at least four reasons why.

1. The pressure to repair a specific problem encouraged a narrow focus on the fault itself and not on its context. In particular, the analysts paid attention to the immediate cause of the failure and not to the underlying design or requirements faults. (how it happened)
2. The fault often had non-obvious **side effects** in places other than the immediate area of the fault.
3. Fixing one problem often caused a failure somewhere else, or the patch addressed the problem in only one place, not in other related places.
4. The fault could not be fixed properly because system functionality or performance would suffer as a consequence.

- Program flaws into two separate logical categories: Inadvertent human errors versus malicious, intentionally induced flaws.
- Types of Flaws

1. Validation error (incomplete or inconsistent): permission checks
2. Domain error: controlled access to data
3. Serialization and aliasing: program flow order
4. Inadequate identification (insufficient) and authentication: basis for authorization
5. Boundary condition violation: failure on first/last case
6. Other exploitable logic errors

NON MALICIOUS PROGRAM ERRORS

- Being human, programmers and other developers make many mistakes, most of which are unintentional and non-malicious.
- Many such errors cause program malfunctions but do not lead to more serious security vulnerabilities.
- **Buffer Overflows**
  - A buffer (or array or string) is a space in which data can be held. A buffer's capacity is finite.

- Suppose a C language program contains the declaration:

```
char sample[10];
```

- Now we execute the statement:

```
sample[10] = 'B';
```

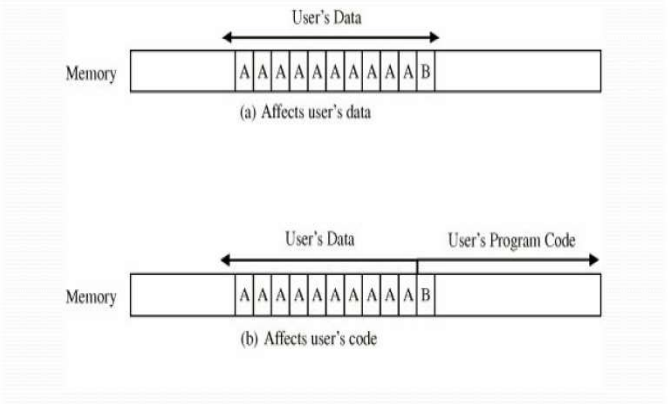
- However, if the statement were

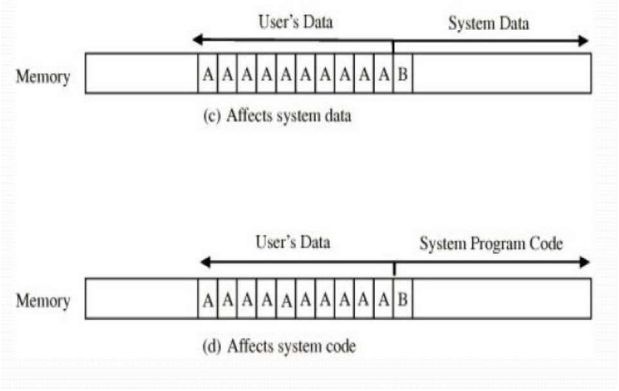
```
sample[i] = 'B';
```

we could not identify the problem until i was set during execution to a too-big subscript.

- Suppose each of the ten elements of the array sample is filled with the letter A and the erroneous reference uses the letter B, as follows:

```
for (i=0; i<=9; i++)
      sample[i] = 'A';
sample[10] = 'B';
```



- Security Implication
  - Two buffer overflow attacks that are used frequently
    1. The attacker may replace code in the system space. By replacing a few instructions right after returning from his or her own procedure, the attacker regains control from the operating system, possibly with raised privileges.
    2. On the other hand, the attacker may make use of the stack pointer or the return register. Subprocedure calls are handled with a stack, a data structure in which the most recent item inserted is the next one removed (last arrived, first served).

- Time-of-Check to Time-of-Use Errors
  - The time-of-check to time-of-use (TOCTTOU) flaw concerns mediation that is performed with a "bait and switch" in the middle. It is also known as a serialization or synchronization flaw.

VIRUSES AND OTHER MALICIOUS CODE

- **Malicious Code Can Do Much (Harm)**
- Malicious code runs under the user's authority.
- Thus, malicious code can touch everything the user can touch, and in the same ways.
- Users typically have complete control over their own program code and data files; they can read, write, modify, append, and even delete them.
- But malicious code can do the same, without the user's permission or even knowledge.

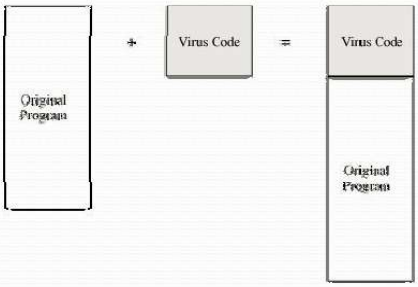
KINDS OF MALICIOUS CODE

- **Malicious code** or **rogue program** is the general name for unanticipated or undesired effects in programs or program parts, caused by an agent intent on damage.
- A **virus** is a program that can replicate itself and pass on malicious code to other non malicious programs by modifying them.
- A **transient virus** has a life that depends on the life of its host
- A **resident virus** locates itself in memory

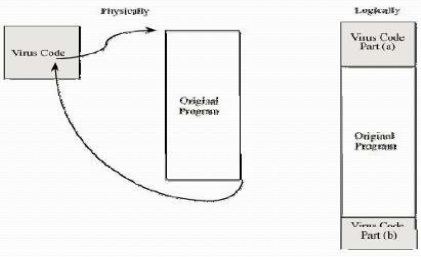
KINDS OF MALICIOUS CODE (CONT'D)

- A **Trojan horse** is malicious code that, in addition to its primary effect, has a second, nonobvious malicious effect
- A **logic bomb** is a class of malicious code that "detonates" or goes off when a specified condition occurs.
- A **time bomb** is a logic bomb whose trigger is a time or date.
- A **trapdoor** or **backdoor** is a feature in a program by which someone can access the program other than by the obvious, direct Call
- A **worm** is a program that spreads copies of itself through a network.
- A **rabbit** is a virus or worm that self-replicates without bound, with the intention of exhausting some computing resource.

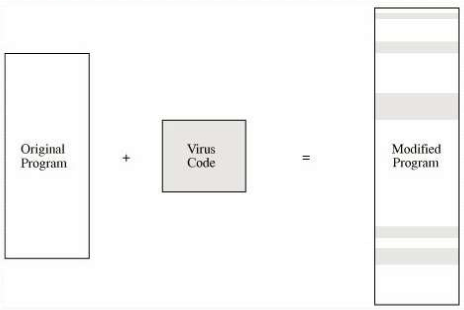
- How Viruses Attach
  - Appended Viruses



- How Viruses Attach (Cont'd)
  - Viruses That Surround a Program

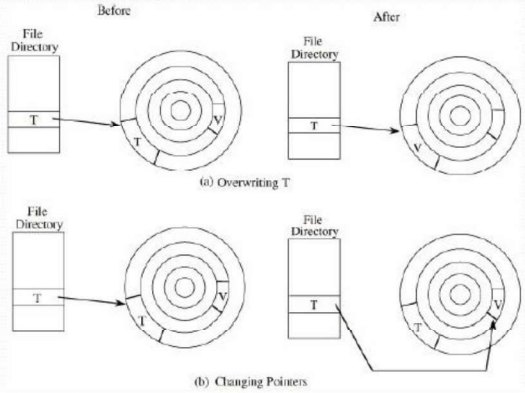


- How Viruses Attach (Cont'd)
  - Integrated Viruses and Replacements

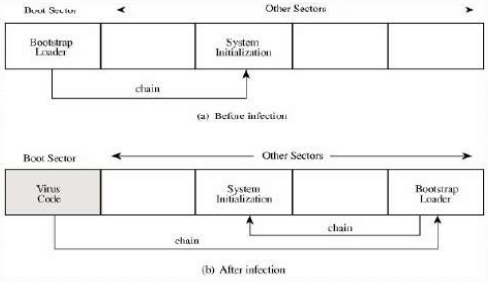


- How Viruses Attach (Cont'd)
  - Document Viruses
    - Implemented within a formatted document, such as a written document, a database, a slide presentation, a picture, or a spreadsheet.

- How Viruses Gain Control

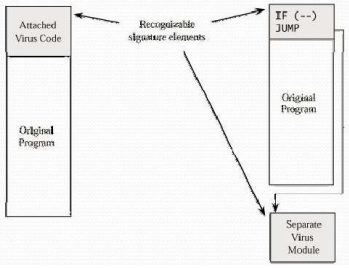


- Homes for Viruses
  - One-Time Execution – the majority of viruses
  - Boot Sector Viruses



- Homes for Viruses (Cont'd)
  - Memory-Resident Viruses
  - Other Homes for Viruses
    - Application programs
    - Libraries
    - Data files – need a startup program

- Virus Signatures
  - Storage Patterns - Most viruses attach to programs that are stored on media such as disks. The attached virus piece is invariant, so the start of the virus code becomes a detectable signature.



- Virus Signatures
  - Execution Patterns - A virus writer may want a virus to do several things at the same time, namely, spread infection, avoid detection, and cause harm.
  - Transmission Patterns - A virus is effective only if it has some means of transmission from one location to another.

- Polymorphic Viruses
  - A virus that can change its appearance
  - Encrypting viruses
    - Uses encryption under various keys to make the stored form of the virus different.
    - Contain three distinct parts:
      - a decryption key,
      - the (encrypted) object code of the virus
      - the (unencrypted) object code of the decryption routine.

Prevention of Virus Infection

The only way to prevent the infection of a virus is not to receive executable code from an infected source.

Prevention of Virus Infection (Cont'd)

- Several techniques for building a reasonably safe community for electronic contact, including the following:
- Use only commercial software acquired from reliable, well-established vendors.
- Test all new software on an isolated computer.
- Open attachments only when you know them to be safe
- Make a recoverable system image and store it safely.
- Make and retain backup copies of executable system files.
- Use virus detectors (often called virus scanners) regularly and update them daily.

TARGETED MALICIOUS CODE

- **Trapdoors** - an undocumented entry point to a module
- **Examples:** A system is composed of modules or components. Programmers first test each small component of the system separate from the other components, in a step called **unit testing**, to ensure that the component works correctly by itself.
- Then, developers test components together during **integration testing**, to see how they function as they send messages and data from one to the other.

CAUSES OF TRAPDOORS

- ***Forget*** to remove them
- Intentionally leave them in the program for ***testing***
- Intentionally leave them in the program for ***maintenance*** of the finished program, or
- Intentionally leave them in the program as a ***covert means of access*** to the component after it becomes an accepted part of a production system

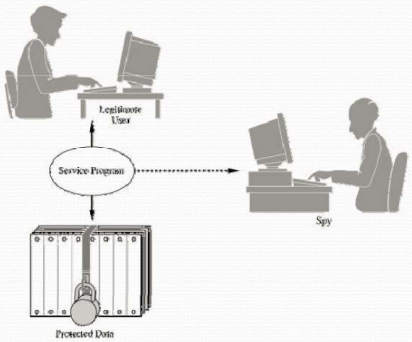
Salami Attack

- Merges bits of seemingly inconsequential data to yield powerful results.

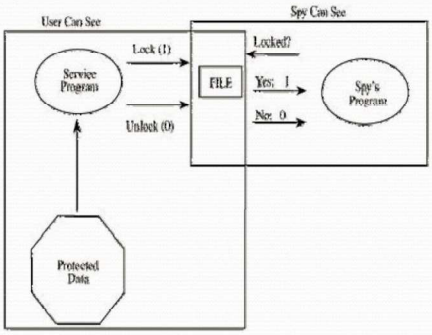
- **Why Salami Attacks Persist**
- Computer computations are notoriously subject to small errors involving rounding and truncation, especially when large numbers are to be combined with small ones.

- **Privilege Escalation -** a means for malicious code to be launched by a user with lower privileges but run with higher privileges.
- **Interface Illusions -** a spoofing attack in which all or part of a web page is false.
- **Keystroke Logging -** retains a surreptitious copy of all keys pressed.
- **Man-in-the-Middle Attacks -** interjects itself between two other programs

- Covert Channels: Programs That Leak Information



- Storage Channels - pass information by using the presence or absence of objects in storage.



CONTROLS AGAINST PROGRAM THREATS

Three types of controls:

- Developmental
- Operating system
- Administrative

CONTROLS AGAINST PROGRAM THREATS

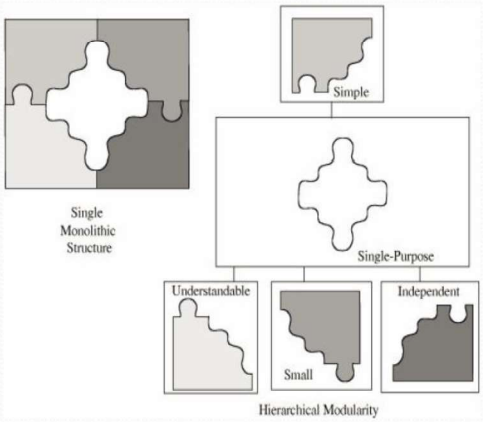
Developmental Controls

- **The Nature of Software Development**
  - Collaborative effort, involving people with different skill sets who combine their expertise to produce a working product
  - Development requires people who can ***specify***, ***design***, ***implement***, ***test***, ***review***, **document**, **manage**, ***maintain*** the system.

CONTROLS AGAINST PROGRAM THREATS

- **Developmental Controls (Cont…)**
- **Modularity, Encapsulation, and Information Hiding**
- A key principle of software engineering is to create a design or code in small, self-contained units, called **components** or **modules**
- If a component is isolated from the effects of other components, then it is easier to trace a problem to the fault that caused it and to limit the damage the fault causes. This isolation is called **encapsulation**.
- **Information hiding** is another characteristic of modular software.

- Modularization is the process of dividing a task into subtasks.



DEVELOPMENTAL CONTROLS (CONT'D): MODULARIZATION

- The goal is to have each component meet four conditions:

1. ***Single-purpose***: performs one function

2. ***Small***: consists of an amount of information for which a human can readily grasp both structure and content

3. ***Simple***: is of a low degree of complexity so that a human can readily understand the purpose and structure of the module

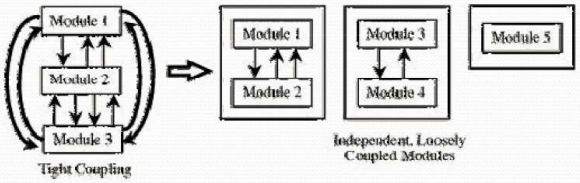
4. ***Independent***: performs a task isolated from other modules

DEVELOPMENTAL CONTROLS (CONT'D): MODULARIZATION

- Several advantages to having small, independent components.
- ***Maintenance***. If a component implements a single function, it can be replaced easily with a revised one if necessary.
- ***Understandability***
- ***Reuse***
- ***Correctness***
- ***Testing***

Developmental Controls (Cont'd): Modularization

- Modularization
  - A modular component usually has high cohesion and low coupling
  - Cohesion, we mean that all the elements of a component have a logical and functional reason for being there.
  - Coupling refers to the degree with which a component depends on other components in the system.



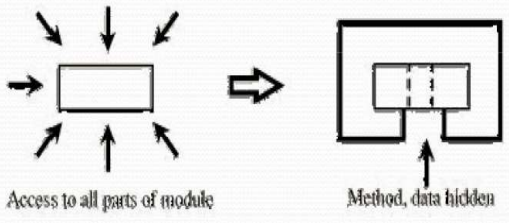
Developmental Controls (Cont'd): Encapsulation

- Encapsulation hides a component's implementation details, but it does not necessarily mean complete isolation
- Berard [BER00] notes that encapsulation is the "*technique for packaging the information [inside a component] in such a way as to hide what should be hidden and make visible what is intended to be visible*".

Developmental Controls (Cont'd) Information Hiding

- Think of a component as a kind of black box, with certain well-define inputs and outputs and a well-defined function.
- Other components' designers do not need to know *how* the module completes its function; it is enough to be assured that the component performs its task in some correct manner.
- This concealment is the information hiding.
- Information hiding is desirable because developers cannot easily and maliciously alter the components of others if they do not know how the components work.

- Developmental Controls (Cont'd)
  - Information Hiding (Cont'd)



- **Developmental Controls (Cont'd)**
- **Mutual Suspicion**
- Mutually suspicious programs operate as if other routines in the system were malicious or incorrect.
- A calling program cannot trust its called sub-procedures to be correct, and a called sub-procedure cannot trust its calling program to be correct.
- Each protects its interface data so that the other has only limited access.

- **Development Controls (Cont'd)**
- Pfleeger et al. [PFL01] recommend several key techniques for building what they call "solid software":
- peer reviews
- hazard analysis
- testing
- good design
- prediction
- static analysis
- configuration management
- analysis of mistakes

- **Development Controls (Cont'd)**
- Peer review (Cont'd)
- A wise engineer who finds a fault can deal with it in at least three ways:

1. by learning how, when, and why errors occur
2. by taking action to prevent mistakes
3. by scrutinizing products to find the instances and effects of errors that were missed

- **Development Controls (Cont'd)**
- **Hazard analysis:**

	Known Cause	Unknown Cause
Known effect	Description of system behavior	Deductive analysis, including fault tree analysis
Unknown effect	Inductive analysis, including failure modes and effects analysis studies	Exploratory analysis, including hazard and operability

- **Developmental Controls (Cont'd)**
- **Confinement**
- A **confined** program is strictly limited in what system resources it can access. If a program is not trustworthy, the data it can access are strictly limited.
- **Genetic Diversity**
- Tight integration of products is a concern.
- A vulnerability in one of these can also affect the others. Fixing a vulnerability in one can have an impact on the others.

- **Development Controls (Cont'd)**
- Peer review
- ***Review:*** The artifact is presented informally to a team of reviewers; the goal is consensus and buy-in before development proceeds further.
- ***Walk-through:*** The artifact is presented to the team by its creator, who leads and controls the discussion. Here, education is the goal, and the focus is on learning about a single document.
- ***Inspection***: The artifact is checked against a prepared list of concerns. The creator does not lead the discussion, and the fault identification and correction are often controlled by statistical measurements.

- **Development Controls (Cont'd)**
- **Hazard analysis**
- A set of systematic techniques intended to expose potentially hazardous system states.
- Usually involves developing hazard lists, as well as procedures for exploring "**what if**" scenarios to trigger consideration of nonobvious hazards.
- A variety of techniques support the identification and management of potential hazards. Among the most effective are
  - **Hazard and operability studies (HAZOP)**
  - **Failure modes and effects analysis (FMEA)**
  - **Fault tree analysis (FTA)**

- **Development Controls (Cont'd)**
- **Testing**
- A process activity that homes in on product quality: making the product failure free or failure tolerant. Usually involves several stages.
- **Unit testing** is done test team can feed a predetermined set of data to the component being tested and observe what output actions and data are produced.
- **Integration testing** is the process of verifying that the system components work together as described in the system and program design specifications.
- A **function test** evaluates the system to determine whether the functions described by the requirements specification are actually performed by the integrated system.

- **Developmental Controls (Cont'd)**
- **Testing (Cont'd)**
- A **performance test** compares the system with the remainder of these software and hardware requirements.
- An **acceptance test**, in which the system is checked against the customer's requirements description.
- A final **installation test** is run to make sure that the system still functions as it should.
- After a change is made to enhance the system or fix a problem, **regression testing** ensures that all remaining functions are still working and that performance has not been degraded by the change.

- **Developmental Controls (Cont'd)**
- **Testing (Cont'd)**
- **Black-box testing** treats a system or its components as black boxes, testers cannot "see inside" the system
- **Clear-box testing** (a.k.a. **white box**). - testers can examine the design and code directly, generating test cases based on the code's actual construction.

- **Developmental Controls (Cont'd)**
- **Good Design**
- Designers should try to anticipate faults and handle them in ways that minimize disruption and maximize safety and security.
- **Passive fault detection** - construct the system so that it reacts in an acceptable way to a failure's occurrence.
- **Active fault detection** - adopting a philosophy of mutual suspicion. Instead of assuming that data passed from other systems or components are correct, we always check that the data are within bounds and of the right type or format.
- We can also use **redundancy**, comparing the results of two or more processes to see that they agree, before we use their result in a task.

- **Developmental Controls (Cont'd)**
- **Static Analysis** - examine its design and code to locate and repair security flaws before a system is up and running

- several aspects of the design and code:
- **control flow structure** - the sequence in which instructions are executed, including iterations and loops.
- **data flow structure** - follows the trail of a data item as it is accessed and modified by the system
- **data structure** - the way in which the data are organized, independent of the system itself.

- **Developmental Controls (Cont'd)**
- **Configuration Management -** the process by which we control changes during development and maintenance
- It is important to know who is making which changes to what and when:
- ***corrective changes***: maintaining control of the system's day-to-day functions
- ***adaptive changes***: maintaining control over system modifications
- ***perfective changes***: perfecting existing acceptable functions
- ***preventive changes***: preventing system performance from degrading to unacceptable levels

- **Developmental Controls (Cont'd)**
- Four activities are involved in configuration management:
- **Configuration identification**
- **Configuration control and change management**
- **Configuration auditing**
- **Status accounting**

- **Developmental Controls (Cont'd)**

- **Configuration identification**
- Sets up baselines to which all other code will be compared after changes are made that is building and document an inventory of all components that comprise the system.
- "Freeze" the baseline and carefully control what happens to it. When a change is proposed and made, it is described in terms of how the baseline changes.

- **Developmental Controls (Cont'd)**
- **Configuration control and configuration management** - ensure we can coordinate separate, related versions.

- Three ways to control the changes
- **Separate files** - have different files for each release or version.
- **Delta** - designate a particular version as the main version of a system and then define other versions in terms of what is different.
- **Conditional compilation**, whereby a single code component addresses all versions, relying on the compiler to determine which statements to apply to which versions.

- **Developmental Controls (Cont'd)**
- **Configuration Management:**
- A **configuration audit** confirms that the baseline is complete and accurate, that changes are recorded, that recorded changes are made, and that the actual software is reflected accurately in the documents.

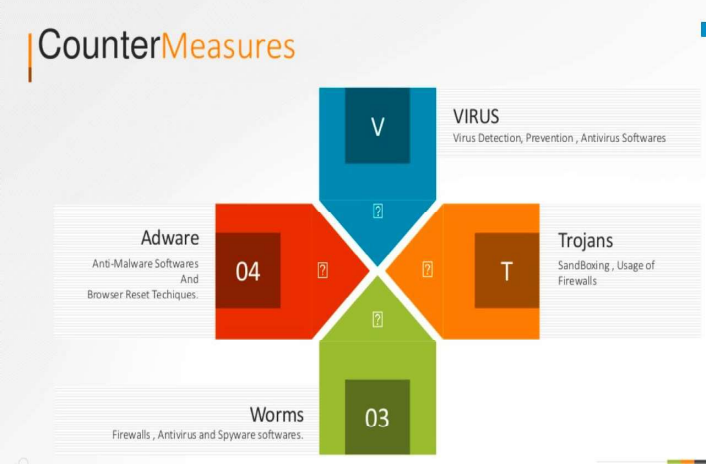
- **Status accounting:**
- Finally, **status accounting** records information about the components: where they came from (for instance, purchased, reused, or written from scratch), the current version, the change history, and pending change requests.

- **Developmental Controls (Cont'd)**
- **Proofs of Program Correctness**
- **Program verification** can demonstrate formally the "correctness" of certain specific programs.
- Making initial the inputs and then checking to see if the desired output is generated.
- Each program statement is translated into a logical description
- Correctness proofs depend on a programmer or logician to translate a program's statements into logical implications.
- The current state of program verification is less well developed than code production.

- **Developmental Controls (Cont'd)**
- **Operating System:**
- Providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.
- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

Virus Countermeasures

- viral attacks exploit lack of integrity control on systems
- to defend need to add such controls
- typically by one or more of:
  - **prevention** - block virus infection mechanism
  - **detection** - of viruses in infected system
  - **reaction** - restoring system to clean state



Difference Between Virus,Worms & Trojans

	VIRUS	WORMS	TROJANS
Definition	A ***computer*** virus attaches itself it travels to a **program** or **file** enabling it to spread from one computer to another, leaving infections	A computer WORM is a self-contained program (or set of programs), that is able to spread functional copies of itself or its segments to other computer systems (usually via network connections).	A **Trojan**, is software that appears to perform a desirable function for the user prior to run or install, but steals information or harms the system

TYPES	VIRUS	WORMS	TROJANS
	1.Trojan Horse 2.Worm 3.Macro	1. "host computer worms" & 2."network worms".	1.Remote Access Trojans 2.Data Sending Trojans 3.Destructive Trojans 4.Proxy Trojans 5.FTP Trojans 6.security software disabler Trojans 7.DoS Trojans

	VIRUS	WORMS	TROJANS
What they do?	virus may exist on your computer but it actually cannot infect your computer unless you run or open the malicious program	the worm consumes too much **system memory** (or **network** bandwidth), causing Web **servers**, network servers and individual computers to stop responding	cause serious damage by deleting files and destroying information on your system.
EXISTENCE	NOT INDEPENDENT	NOT INDEPENDENT	INDEPENDENT

	VIRUS	WORMS	TROJANS
SELF-REPLICATION	. A virus attaches itself to, and becomes part of, another executable program	a worm is self-contained and does not need to be part of another program to propagate itself.	Unlike virus and worms Trojans do not reproduce by infecting other files nor do they self-replicate.

	VIRUS	WORMS	TROJANS
Propagation	virus does not have a propagation vector. i.e., it will only effect one host and does not propagate to other hosts.	Worms propagate and infect other computers.	. Trojans are also known to create a backdoor on your computer that gives malicious users access to your system, possibly allowing confidential or personal information to be compromised

SECURITY ISSUES

- Code Injection: SQL Injection attack
  - Two ways to prevent code injection: avoiding vulnerable code and filtering input. A Web Application Firewall (WAF), updates threat database in real-time to filter application input to protect against code injection
- Data Breach
- Malware Infection
- Distributed Denial of Service Attack(DDoS): involves a group of computers being harnessed together by a hacker to flood the target with traffic.
  - DDoS Protection
- Malicious Insiders

History of protection in operating systems

- There were no operating systems: Users entered their programs directly into the machine in binary by means of switches
- In many cases, program entry was done by physical manipulation of a toggle switch; in other cases, the entry was performed with a more complex electronic method (keyboard).
- Each user had exclusive use of the computing system, users were required to schedule blocks of time for running the machine.
- These users were responsible for loading their own libraries of support routines assemblers, compilers, shared subprograms and "cleaning up" after use by removing any sensitive code or data.

History

- First operating systems were called **executives**, designed to assist individual programmers and provided linkers and loaders for relocation, easy access to compilers and assemblers, and automatic loading of subprograms from libraries.
- Mo**nitors or** Multiprogramming was implemented. 2 users could interleave access to the resources of a single computing system, researchers developed concepts such as scheduling, sharing, & parallel use.

Protected Objects

- Memory
- Sharable I/O devices, such as disks
- Serially reusable I/O devices (printers)
- Sharable programs and sub procedures
- Networks
- Sharable data

Security Methods of OS

- **Hardware-Enforced Protection**
  - *Physical separation*
  - *Temporal separation*
  - *Logical separation*
  - *Cryptographic separation*

Separation and Sharing

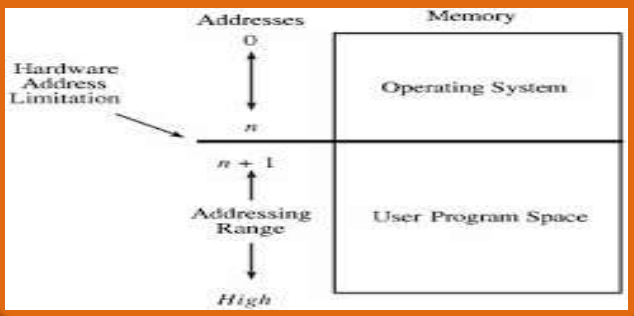
Operating systems can support separation and sharing.

- *Do not protect*
- *Isolate*
- Share all or share nothing
- Share via access limitation
- Share by capabilities
- Limit use of an object

Memory and Address Protection

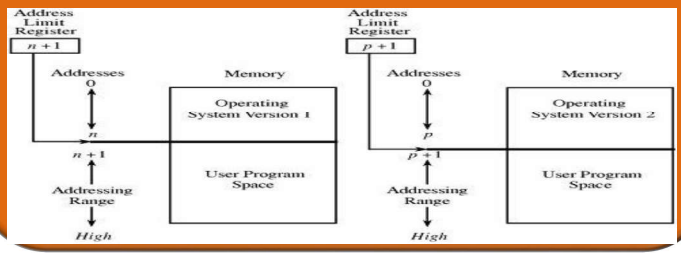
- Protection can be built into hardware mechanisms that control efficient use of memory, so solid protection can be provided at essentially no additional cost
- Fence
- Relocation
- Base/Bounds Registers
- Tagged Architecture
- Paging
- Combined Paging with Segmentation

Memory and Address Protection



Memory and Address Protection: Fence

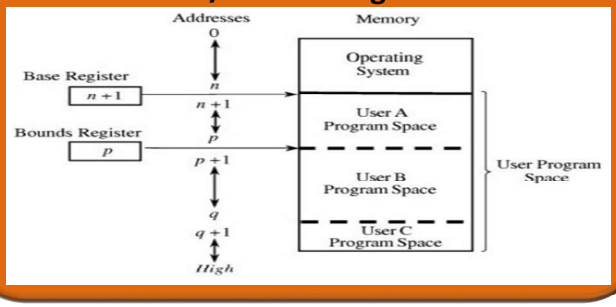
Fence – simple form of memory protection for single OS user.
Fence Register (hardware register) containing the address of the end of OS. One way protection only(a user from OS, not a user from other user)



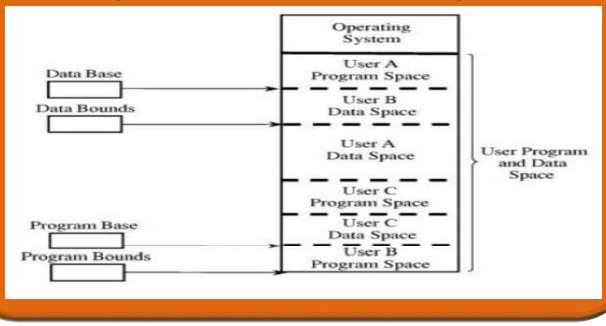
Memory and Address Protection: Relocation

- **Relocation** is the process of taking a program written as if it began at address 0 and changing all addresses to reflect the actual address at which the program is located in memory.
  - adding a constant
- Relocation factor to each address of the program. That is, the relocation factor is the starting address of the memory assigned for the program

Memory and Address Protection: Base/Bounds Registers



Memory and Address Protection: Two pairs of Base/Bounds Registers



Memory and Address Protection: Tagged Architecture

- Another problem with using base/bounds registers for protection or relocation is their contiguous nature.
- Each pair of registers confines accesses to a consecutive range of addresses.
- A compiler or loader can easily rearrange a program so that all code sections are adjacent and all data sections are adjacent.
- An alternative is **tagged architecture**, in which every word of machine memory has one or more extra bits to identify the access rights to that word. The bits are tested every time an instruction accesses that location.

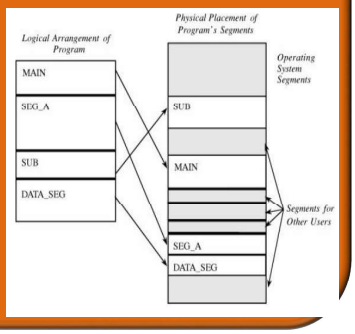
Memory and Address Protection: Tagged Architecture

Tag	Memory Word
R	0001
RW	0137
R	0099
X	
X	
X	
X	
X	
X	
R	4091
RW	0002

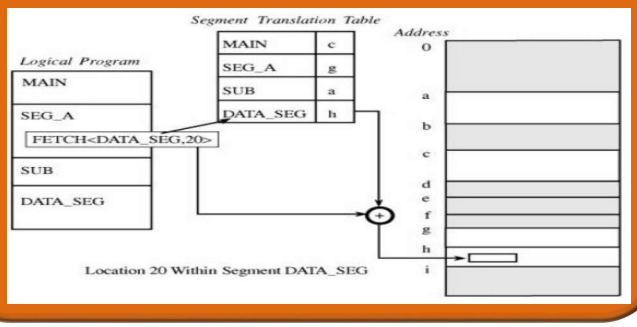
Code: R = Read-only RW = Read/Write
X = Execute-only

Memory and Address Protection: Segmentation

- Each segment has a unique name. A code or data item within a segment is addressed as the pair <name, offset>, where name is the name of the segment containing the data item and offset is its location within the segment.



Memory and Address Protection: Segmentation



Memory and Address Protection: Segmentation

Segmentation advantages for the operating system:

- The operating system can place any segment at any location or move any segment to any location, even after the program begins to execute. Operating system needs only update the address in that one table when a segment is moved.
- A segment can be removed from main memory (and stored on an auxiliary device) if it is not being used currently.
- Every address reference passes through the operating system, so there is an opportunity to check each one for protection.

Memory and Address Protection: Segmentation

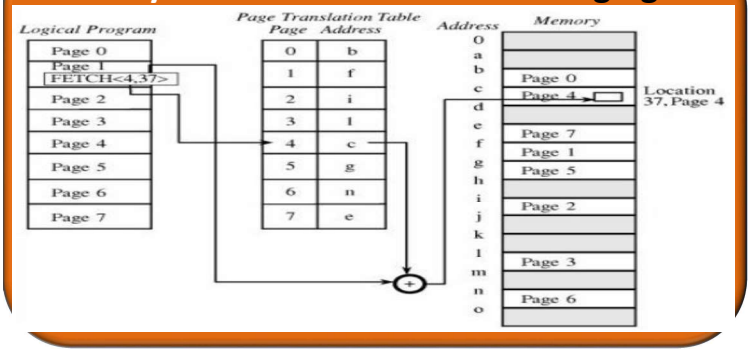
Segmentation offers these security benefits:

- Each address reference is checked for protection.
- Many different classes of data items can be assigned different levels of protection.
- Two or more users can share access to a segment, with potentially different access rights.
- A user cannot generate an address or access to an unpermitted segment.

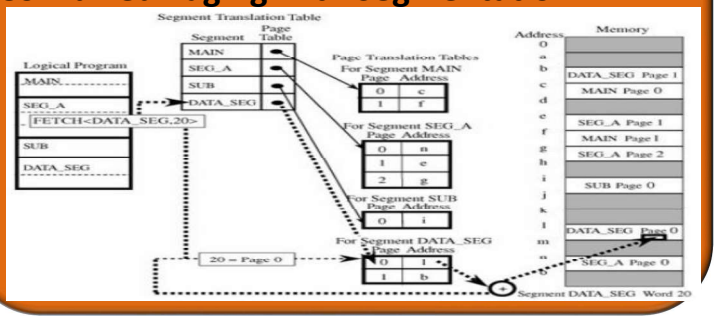
Memory and Address Protection: Paging

- One alternative to segmentation is **paging**.
- The program is divided into equal-sized pieces called pages, and memory is divided into equal-sized units called **page frames**.
- As with segmentation, each address in a paging scheme is a two-part object, this time consisting of <page, offset>.
- Each address is again translated by a process similar to that of segmentation: The OS maintains a table of user page numbers and their true addresses in memory. The page portion of every <page, offset> reference is converted to a page frame address by a table lookup; the offset portion is added to the page frame address to produce the real memory address of the object referred to as <page, offset>.

Memory and Address Protection: Paging



Memory and Address Protection: Combined Paging with Segmentation



Control of Access to General Objects

Protecting memory is a specific case of the more general problem of protecting objects.

- Memory
- A file or data set on an auxiliary storage device
- An executing program in memory
- A directory of files
- A hardware device
- A data structure, such as a stack
- A table of the operating system
- Instructions, especially privileged instructions
- Passwords and the user authentication mechanism
- The protection mechanism itself