

CYBER SECURITY

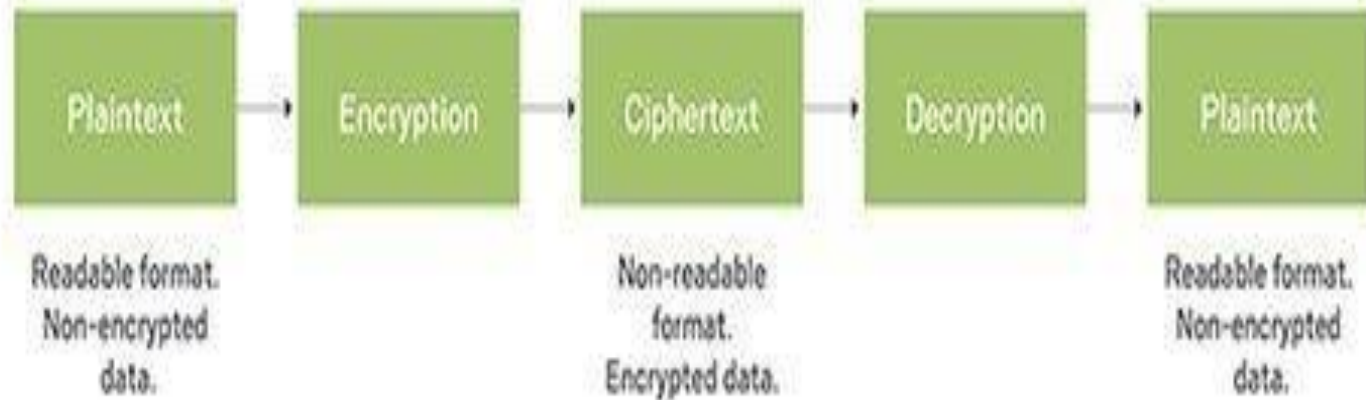
CRYPTOGRAPHY

Important Terms

- **Plain Text** - Plain text is the original message or data that is readable and understandable by humans.
- **Cipher Text** - Cipher text is the encrypted form of plain text that looks random and unreadable to anyone who doesn't have the decryption key
- **Cryptography** – Cryptography is the science and art of securing information by converting it into an unreadable format (cipher text), so only authorized users can access it (combination of encryption and decryption).
- **Cryptanalysis** – Cryptanalysis is the study and practice of breaking cryptographic systems, i.e., trying to decode a message without having access to the key.
- It is used:
 - By attackers to crack codes.
 - By researchers to test the strength of encryption methods.

- **Cryptology** – Combination of Cryptography and Cryptanalysis
- **Encryption** – Process of converting readable text into unreadable text
- **Decryption** - Process of converting unreadable text into readable text
- **Key** - A string of characters, often a large number, that is used in conjunction with a cryptographic algorithm to encrypt or decrypt data.

Cryptography



Encryption techniques

```
graph TD; A[Encryption techniques] --> B[Substitution]; A --> C[Transposition]; B --> D[Monoalphabetic]; B --> E[Polyalphabetic]; D --> F[Ceaser Cipher]; E --> G[Vigenère cipher]; C --> H[Rail fence technique<br/>simple columnar<br/>Vernam Cipher];
```

Substitution

Monoalphabetic

Ceaser Cipher

Polyalphabetic

Vigenère cipher

Transposition

Rail fence technique
simple columnar
Vernam Cipher

Substitution techniques

Substitution techniques in cryptography involve replacing units of plaintext (like letters or bits) with ciphertext units, according to a fixed rule or key.

- **Monoalphabetic cipher**

Where each letter in the plaintext is replaced by a fixed different letter throughout the message

Ex – A is replaced by J in whole msg

- **Polyalphabetic cipher**

Uses multiple substitution alphabets to encrypt the data.

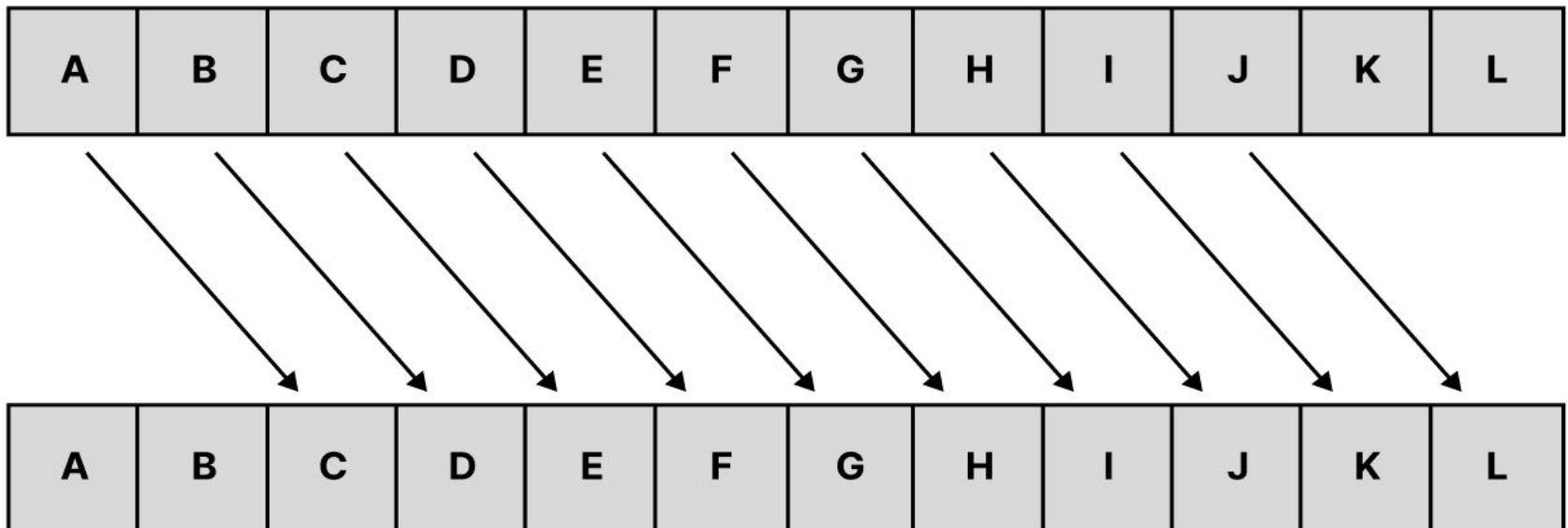
Ex- A is replaced by multiple characters

Caesar Cipher (Monoalphabetic Cipher)

- The Caesar cipher works on the basic principle of substitution where each letter of a text is replaced by a letter occurring at a fixed number of positions in the alphabet.

This fixed number is referred to as a “shift” or “key”

K = 2 **Shifts the alphabet 2 characters to the right**



Example:

- Plain text – HELLO key-3
- Cipher text - KHOOR

- Plain text – “Nature is the art of God” Key – 4
- Cipher text - Rexyvi mw xli evx sj Ksh

- Plain text - Nature is the source of all true knowledge
- Key- 5
- Cipher text - Sfyzwj nx ymj xtzwhj tk fqq ywzj pstbqjilj

Vigenère Cipher

Plaintext: ATTACKATDAWN

Key: LEMON

The keyword is **repeated** to match the length of the plaintext:

Plaintext: A T T A C K A T D A W N

Keyword: L E M O N L E M O N L E

Ciphertext: **LXFOPVEFRNHR**

Plaintext

Key

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

Example

• **Plaintext:** HELLO **Key:** KEY

• **Plaintext:** H E L L O

• **Keyword:** K E Y K E

• **Ciphertext:** **RIJVS**

• **Plaintext:** Nature is the art of God

• **Key:** RADHA

Plaintext: N A T U R E I S T H E A R T O F G O D

Keyword: R A D H A R A D H A R A D H A R A D H

• **Ciphertext:** **EAWBRVIVAHVAUAOWGRK**

Transposition techniques

- Where the positions of characters in the plaintext are rearranged according to a certain system, without changing the actual letters.

Transposition ciphers shuffle the characters around using a defined algorithm or pattern

Example-

- Rail fence technique
- Simple columnar
- Vernam Cipher (One-Time Pad)

Rail fence technique:

write your message in zigzag lines across the page, and then read off each row.

Plain Text: CODESPEEDY
Key= 4

C						E			
	O				P		E		
		D		S				D	
			E						Y

Cipher Text: CEOPEDSDEY

Example

- Plain text - "defend the east wall" key – 3
- Cipher text – **DNETLEEDHESWLFTAA**
- Plain text – code breaking is fun key - 3

C				B				K				I				N
	O		E		R		A		I		G		S		U	
		D				E				N				F		

CODE BREAKING IS FUN

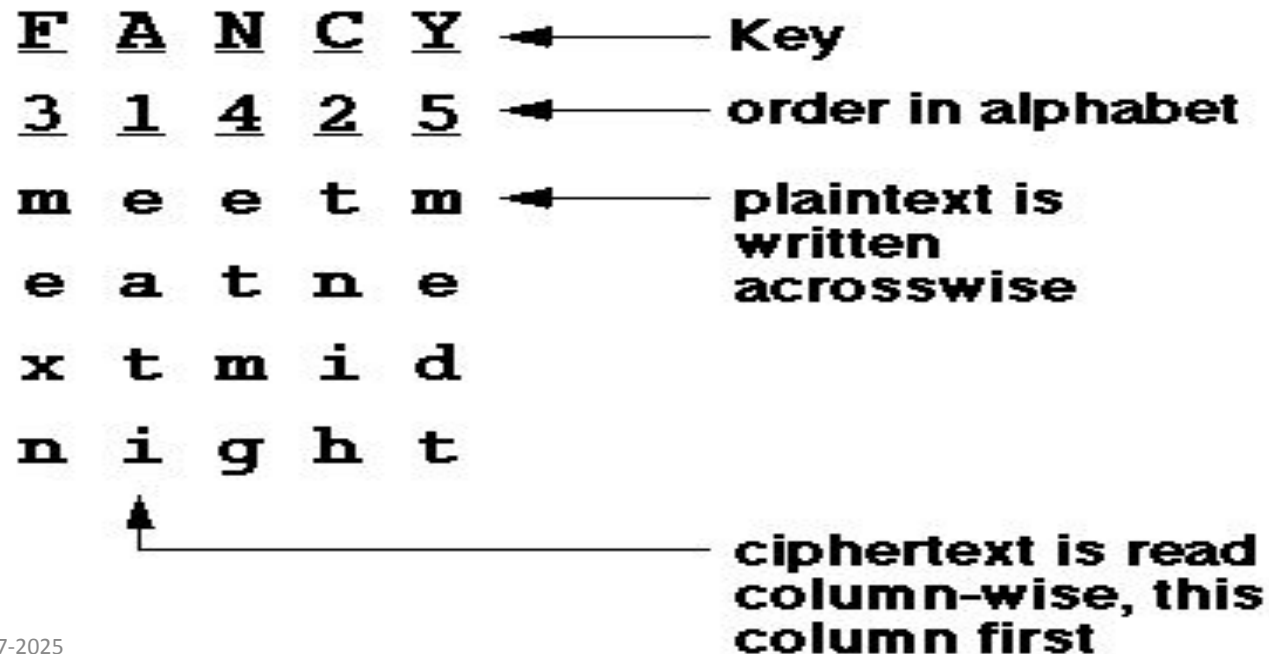
C _ _ _ B _ _ _ K _ _ _ I _ _ _ N →
_ O _ E _ R _ A _ I _ G _ S _ U _ →
_ D _ _ E _ _ _ N _ _ _ F _ _ →

CBKI NOER AIGS UDENF

Simple columnner

The plaintext is written into a grid (or rectangle) row by row. Then, the ciphertext is created by reading the columns in a specific order, determined by a keyword.

- Plain Text – meet me at next mid night
- Key – FANCY cipher text - *EATITNIHMEXNETMGMEDT*



Example:

- Plain Text: TREE IS GREEN, Key = HACK
- Plain Text: This is the plain text to demonstrate transposition technique
- Key – secret
- Plaintext: Nature is the art of God
- Key: 35124

Vernam Cipher (One-Time Pad)

- 1) Plain text character will be represented by the numbers as
A=0, B=1, C=2,... Z=25.
- 2) Add each corresponding number of a plain text message to the
key text alphabet numbers.
- 3) If the sum is greater than or equal to 26, subtract 26 from it.
- 4) Translate each number back to corresponding letters and we
got our cipher text.
- 5) Ciphertext = Plaintext \oplus Key

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Example:

Vernam Cipher

1. Plain text:	H	E	L	L	O	B	O	B
	7	4	11	11	14	1	14	1
	+							
2. One-Time Pad:	U	S	I	F	J	E	V	K
	20	18	8	5	9	4	21	10
3. Total:	27	22	19	16	23	5	35	11
4. If>25,subtract 26:	2	22	19	16	23	5	9	11
5. Cipher text:	C	W	T	Q	X	F	J	L

- Plain Text – INCLUDEHELP
- Key – ATQXRZWOB YV

	I	N	C	L	U	D	E	H	E	L	P
Plain text:	8	13	2	11	20	3	4	7	4	11	15
	0	19	16	23	17	25	22	14	1	24	21
One-time pad:	A	T	Q	X	R	Z	W	O	B	Y	V
<hr/>											
Initial Total:	8	32	18	34	37	28	26	21	5	35	36
Subtract 26, if >25:	8	6	18	8	11	2	0	21	5	9	10
Cipher Text:	I	G	S	I	L	C	A	V	F	J	K

Examples

**1) Plaintext: HELLO
Key: XMCKL**

**2) Plaintext: SECRET
Key: KDWGRU**

**3) Plaintext: GARDEN
Key: CIPHER**

Decryption:

- Reverse the process to get plaintext
- Example:
- Cipher = EQNVZ \rightarrow 4 16 13 21 25
- Key = XMCKL \rightarrow 23 12 2 10 11
- Plain = $(C - K + 26) \bmod 26$
- = $(4 - 23 + 26) = 7 \rightarrow$ H
- = $(16 - 12) = 4 \rightarrow$ E
- = $(13 - 2) = 11 \rightarrow$ L
- = $(21 - 10) = 11 \rightarrow$ L
- = $(25 - 11) = 14 \rightarrow$ O

Substitution Cipher

1. In this, we replace letters of plain text with other letters.

2. It is easy to break.

3. It is easy to implement.

4. It is less resistant to frequency analysis.

5. Ex: – Caesar cipher, Hill cipher, etc.

Transposition Cipher

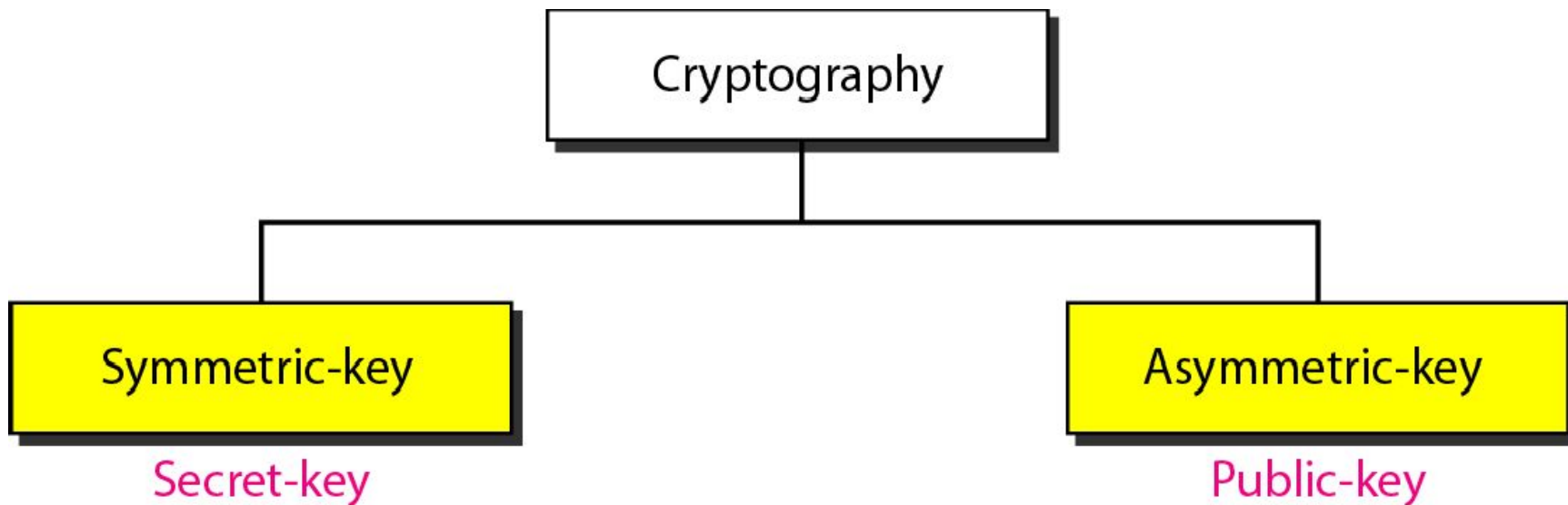
In this, we rearrange the letters of plain text.

It is tough to break.

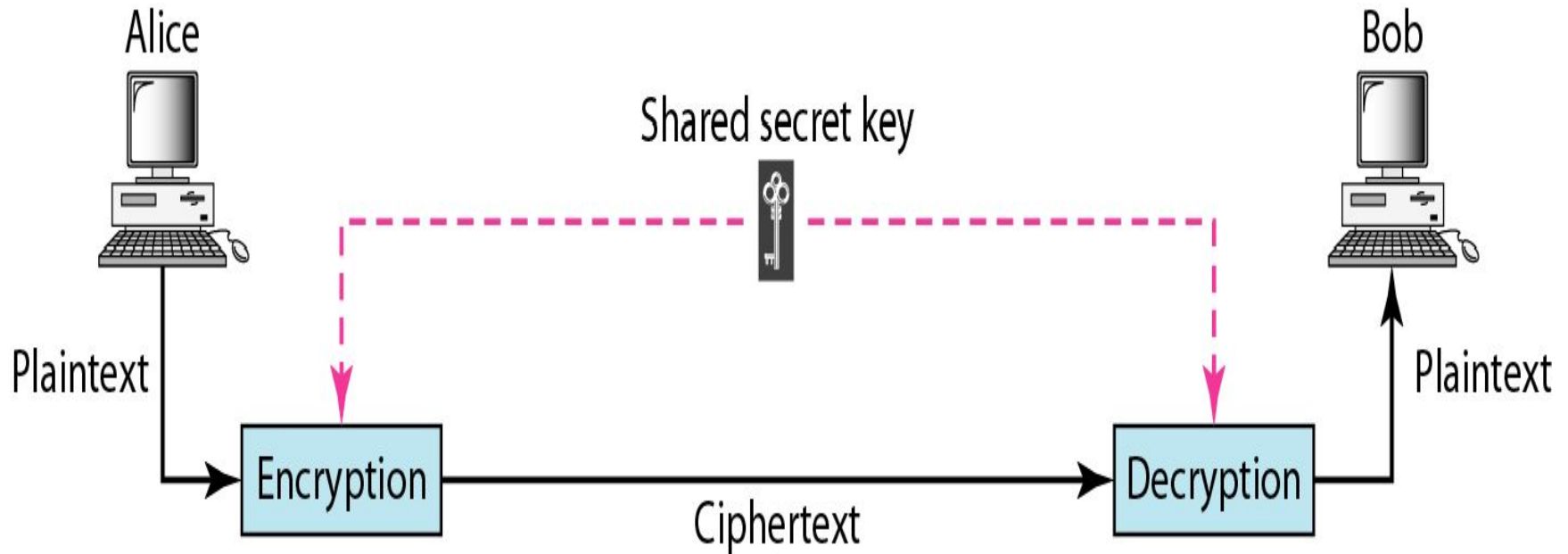
It is hard to implement.

It is more resistant to frequency analysis.

Ex: – Vernam cipher, Rail-fence technique.



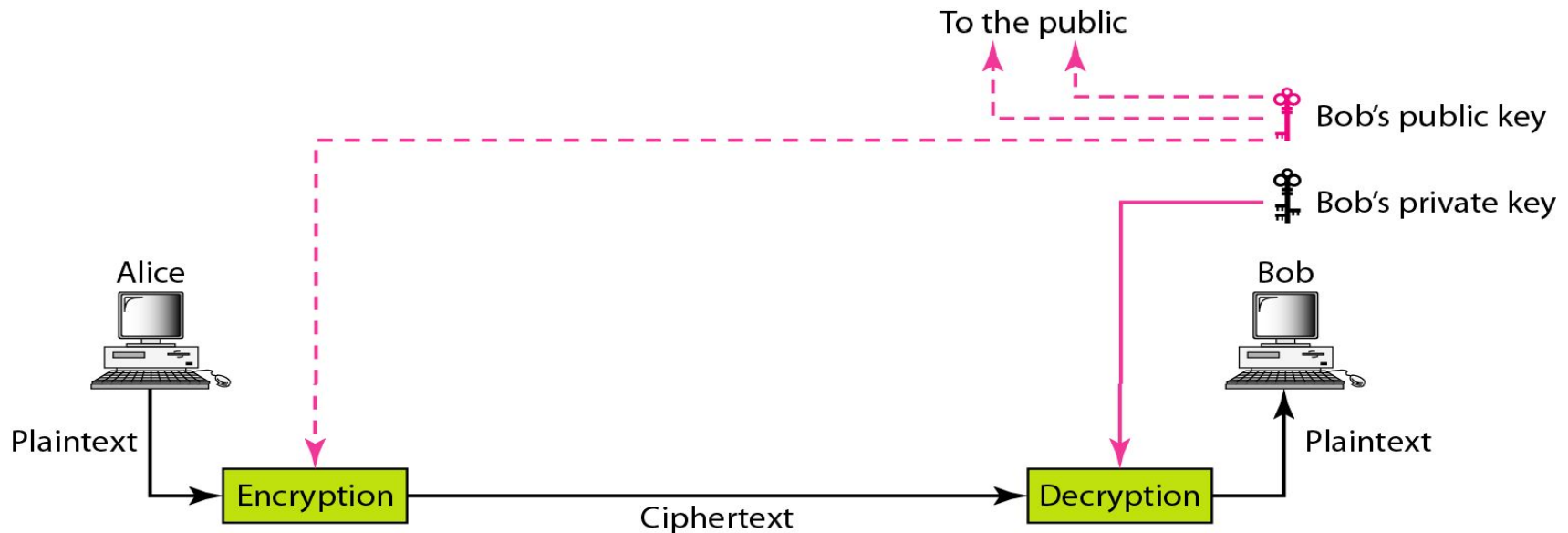
Symmetric-key encryption



Symmetric encryption uses the same key for both encryption and decryption, making it fast but requiring secure key exchange

asymmetric encryption

- Asymmetric encryption employs a pair of keys: a public key for encryption and a private key for decryption, providing enhanced security but with reduced speed



Keys

- secret key - a piece of information used to encrypt and decrypt data. It's the core of symmetric-key cryptography, where the same key is used for both encryption and decryption.
- Public Key
- Private Key



Secret key

Symmetric-key cryptography



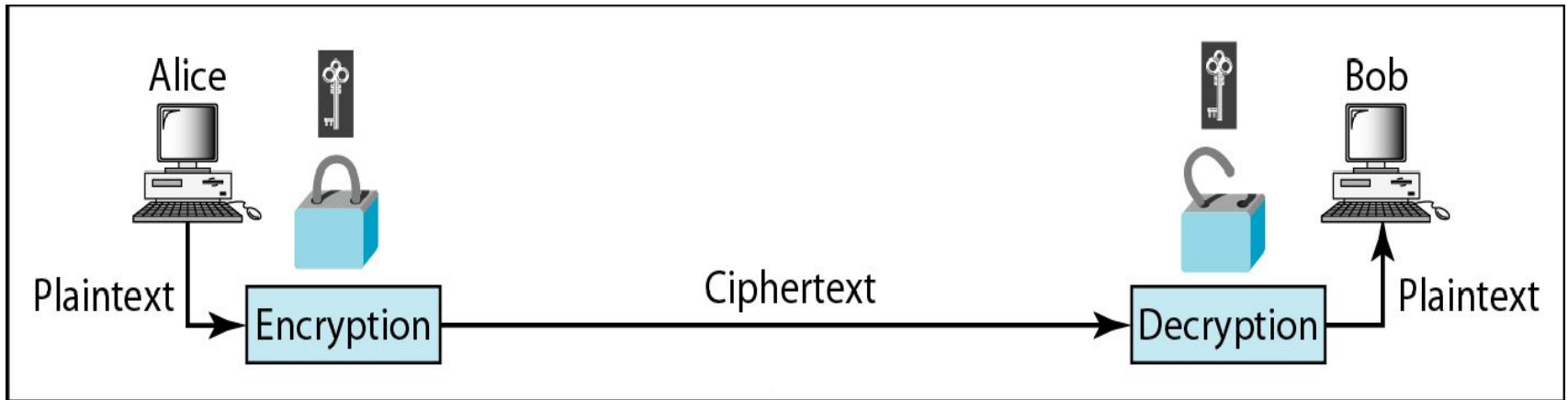
Public key



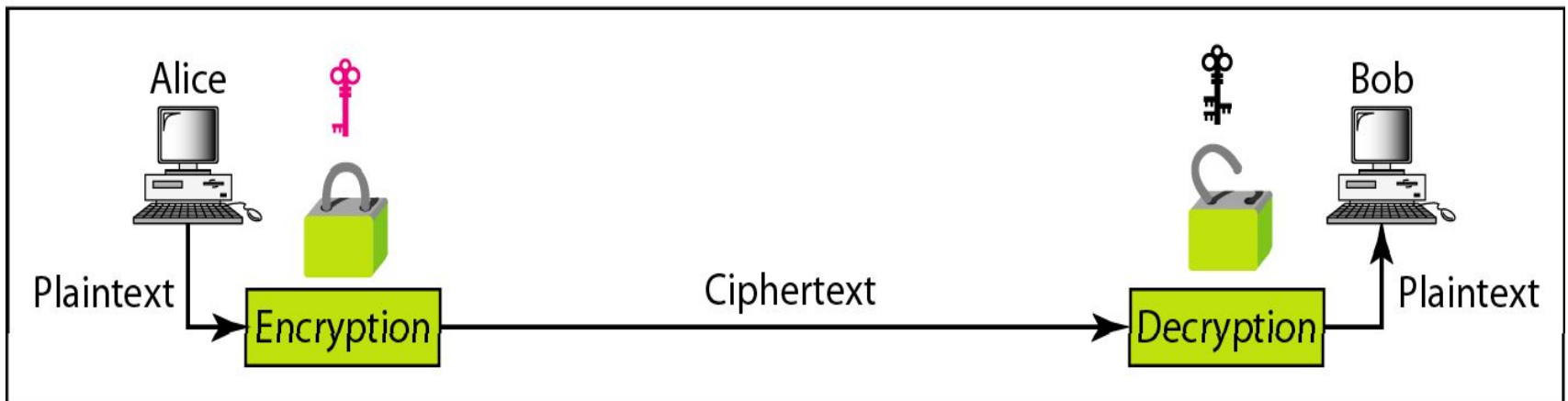
Private key

Asymmetric-key cryptography

Symmetric Key Encryption	Asymmetric Key Encryption
It only requires a single key for both encryption and decryption.	It requires two keys, a public key and a private key, one to encrypt and the other to decrypt.
The encryption process is very fast.	The encryption process is slow.
It is used when a large amount of data needs to be transferred.	It is used to transfer small amount of data.
It only provides confidentiality.	It provides confidentiality, authenticity, and non-repudiation.
In symmetric key encryption, resource utilization is low compared to asymmetric key encryption.	In asymmetric key encryption, resource utilization is high.
It is efficient as it is used for handling large amount of data.	It is comparatively less efficient as it can handle a small amount of data.
Security is lower as only one key is used for both encryption and decryption purposes.	Security is higher as two keys are used, one for encryption and the other for decryption.
Examples: 3DES, AES, DES and RC4	Examples: Diffie-Hellman, ECC, El Gamal, DSA and RSA



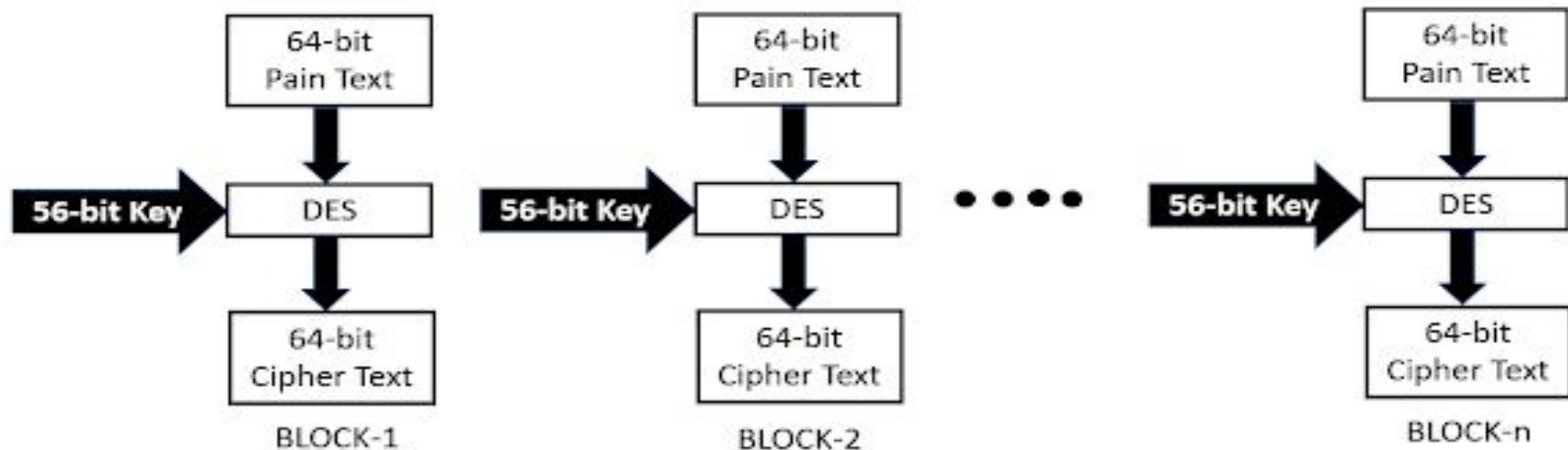
a. Symmetric-key cryptography



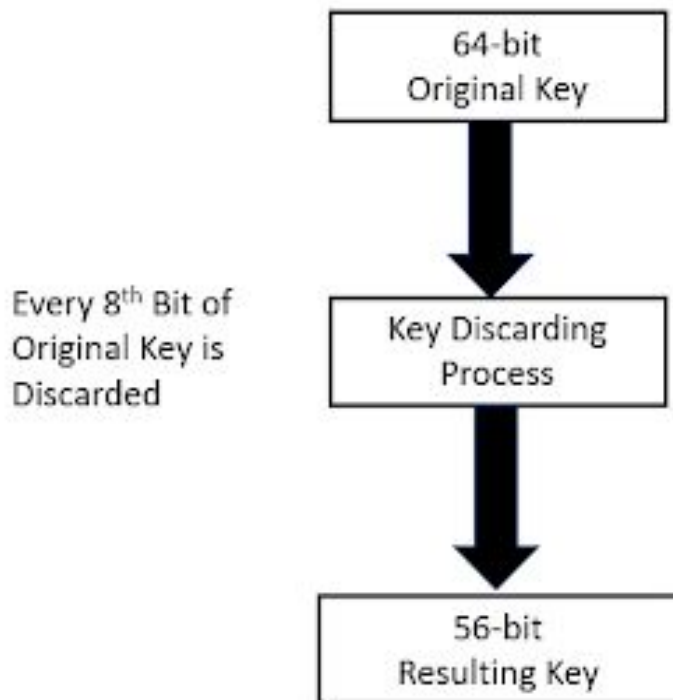
b. Asymmetric-key cryptography

Symmetric encryption Algorithm :DES


- DES is symmetric cipher algorithm and use block cipher method for encryption and decryption.



Key Discarding Process

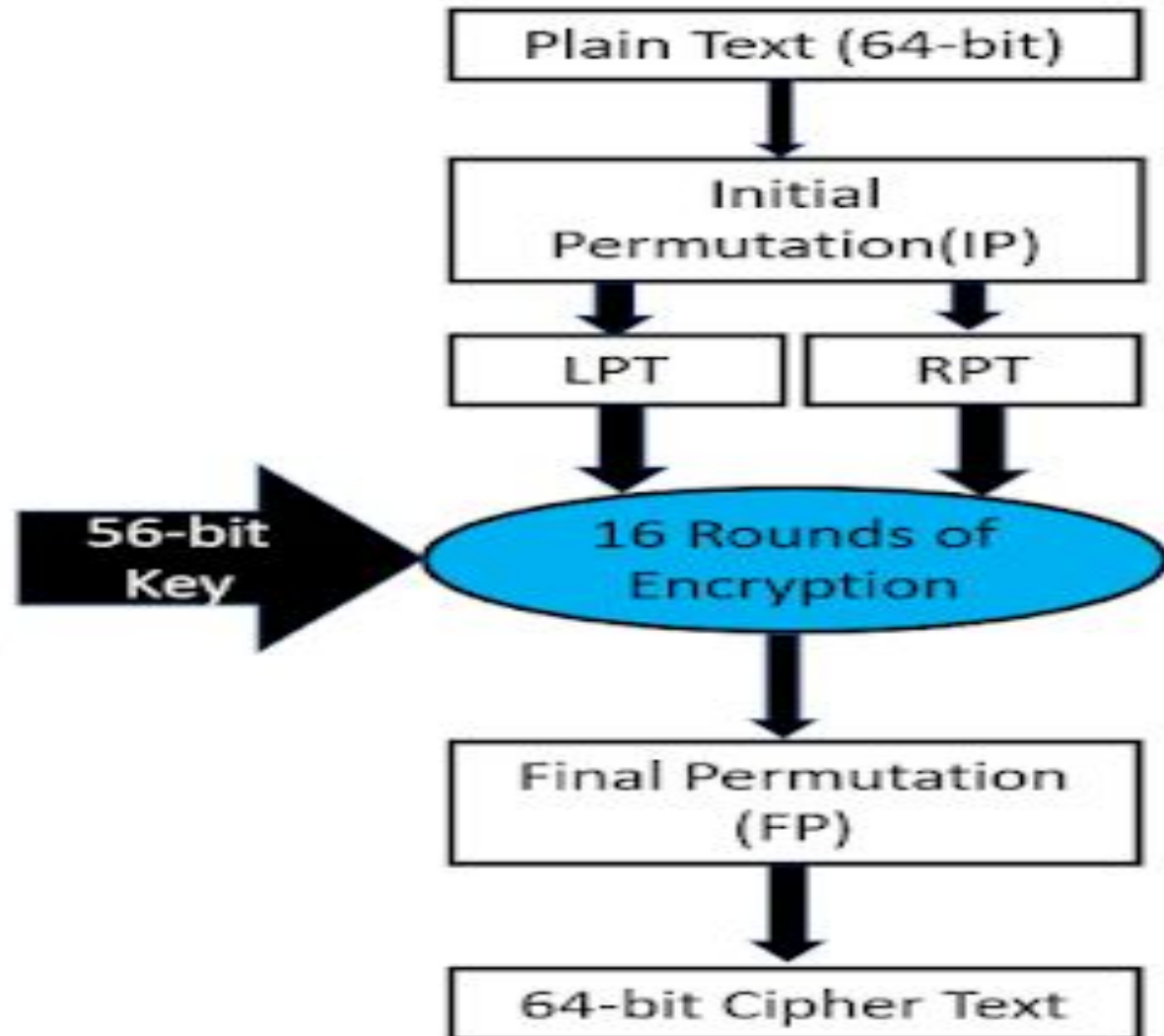


1	2	21	38	58	15	37	26
22	55	44	3	53	27	11	60
49	28	14	42	61	48	63	41
18	39	56	10	64	16	62	8
45	40	20	54	4	33	34	52
7	30	47	59	32	5	35	25
29	12	13	6	24	46	57	36
17	23	50	31	43	51	9	19



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

Steps of DES



Step – 1: 64-bit plain text block is given to Initial Permutation (IP) function.

Step – 2: IP performed on 64-bit plain text block.

Step – 3: IP produced two halves of the permuted block known as Left Plain Text (LPT) and Right Plain Text (RPT).

Step – 4: Each LPT and RPT performed 16-rounds of encryption process.

Step – 5: LPT and RPT rejoined and Final Permutation (FP) is performed on combined block.

Step – 6: 64-bit Cipher text block is generated.

Initial Permutation (IP) & Generate LPT -RPT

Initial Permutation performed only once. Bit sequence have changed as per IP table.

For Example: 1st bit takes 40th Position, 58th bit take 1st position

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

Output of IP is divided into two equal halves known as LPT, RPT. (LPT – 32 bits, RPT – 32 bit)

16 Rounds of Encryption

Step – 1: Key Transformation
(56-bit key)

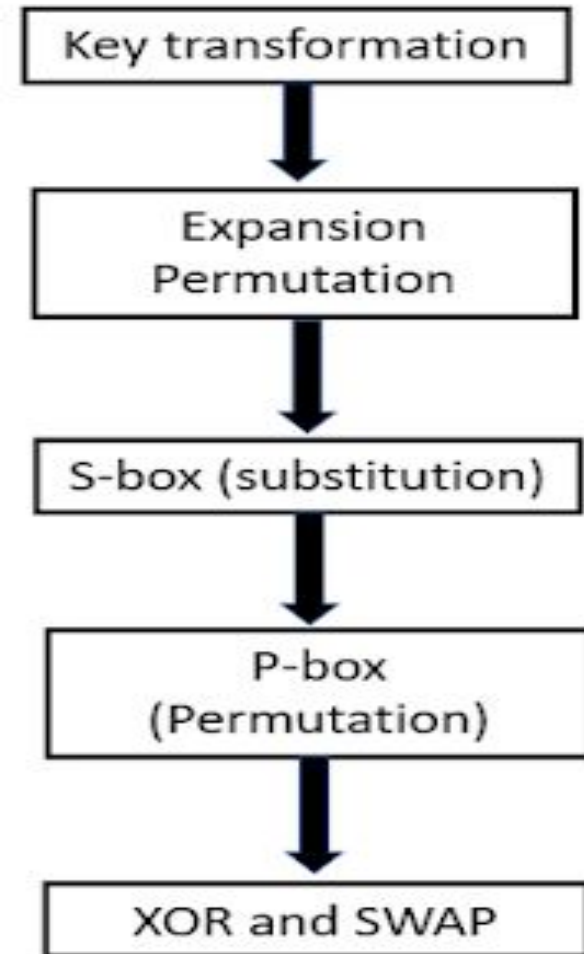
Key Bit Shifted per round
Compression Permutation

Step–2:Expansion
permutation of Plain Text and
X-OR (P.T. size: 48 bit, C.T.
size: 48 bit)

Step – 3: S-box Substitution

Step – 4: P-box (Permutation)

Step – 5: X-OR and Swap.



16 Rounds of Encryption

• Step – 1: Key Bit Shifted per Round

- 56-bit key is divided into two halves each of 28-bits.
- Circular left shift is performed on each half.
- Shifting of Bit position is depending on round.
- For round number 1,2,9 and 16 shifts are done by one position.
- For remaining rounds shift is done by 2 positions.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Key bit shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Compression Permutation

56-bit input with bit shifting position

Generates 48-bit key (Compression of Key bit)

Drop 9, 18, 22, 25, 35, 38, 43 and 54 bits.

Generated 48 bits keys are as below:

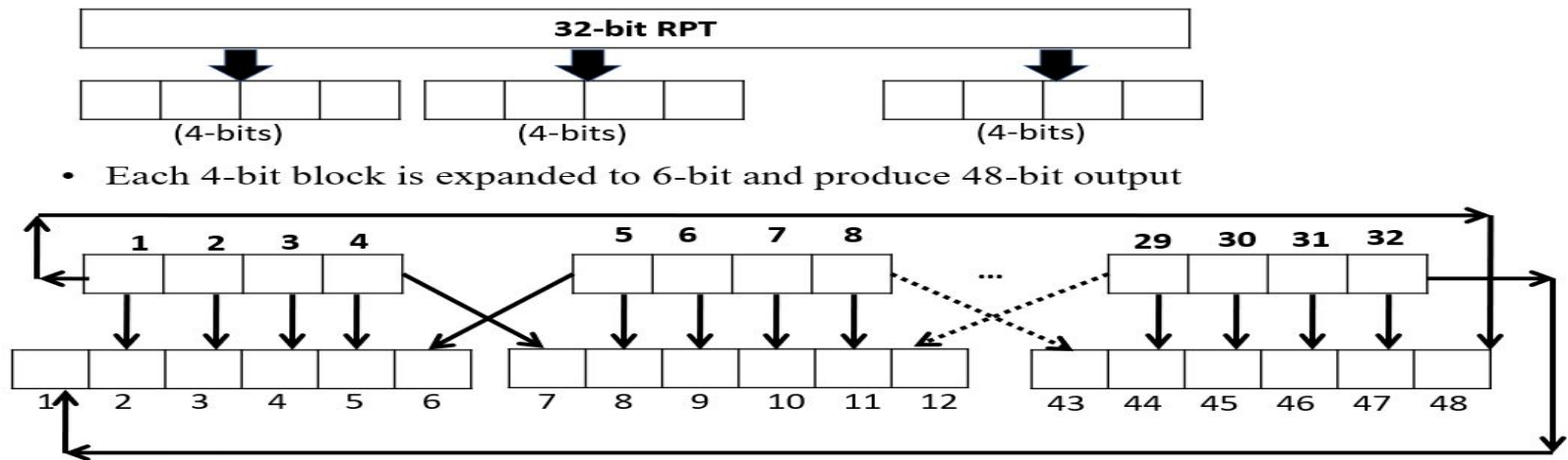
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Step – 2: Expansion Permutation and X-OR

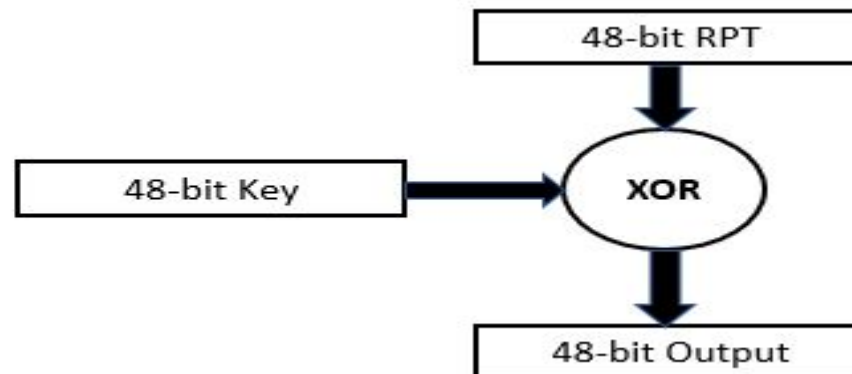
32-bit RPT of IP is expanded to 48-bits

Expansion permutation steps:

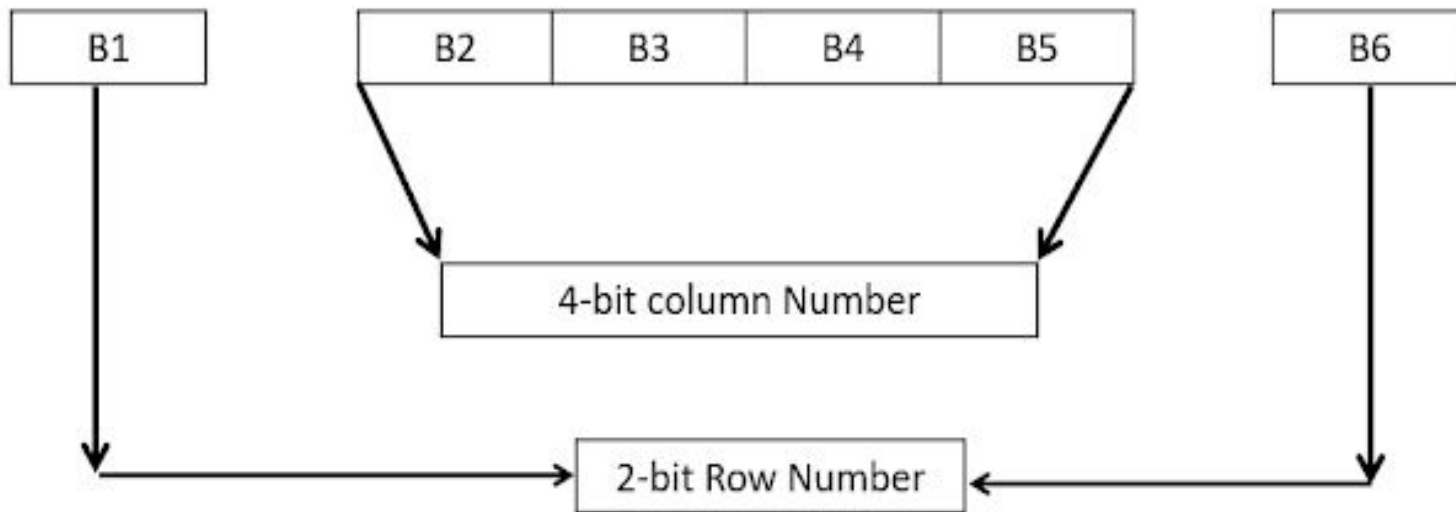
32-bit RPT is divided into 8-blocks each of 4-bits



48-bit RPT is XORed with 48-bit Key and output is given to S-Box.



Step – 3: S-BOX Substitution



S ₅		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Example: 011011 → 1001

Step -4: P-BOX Permutation

Output of s-box is given to p-box

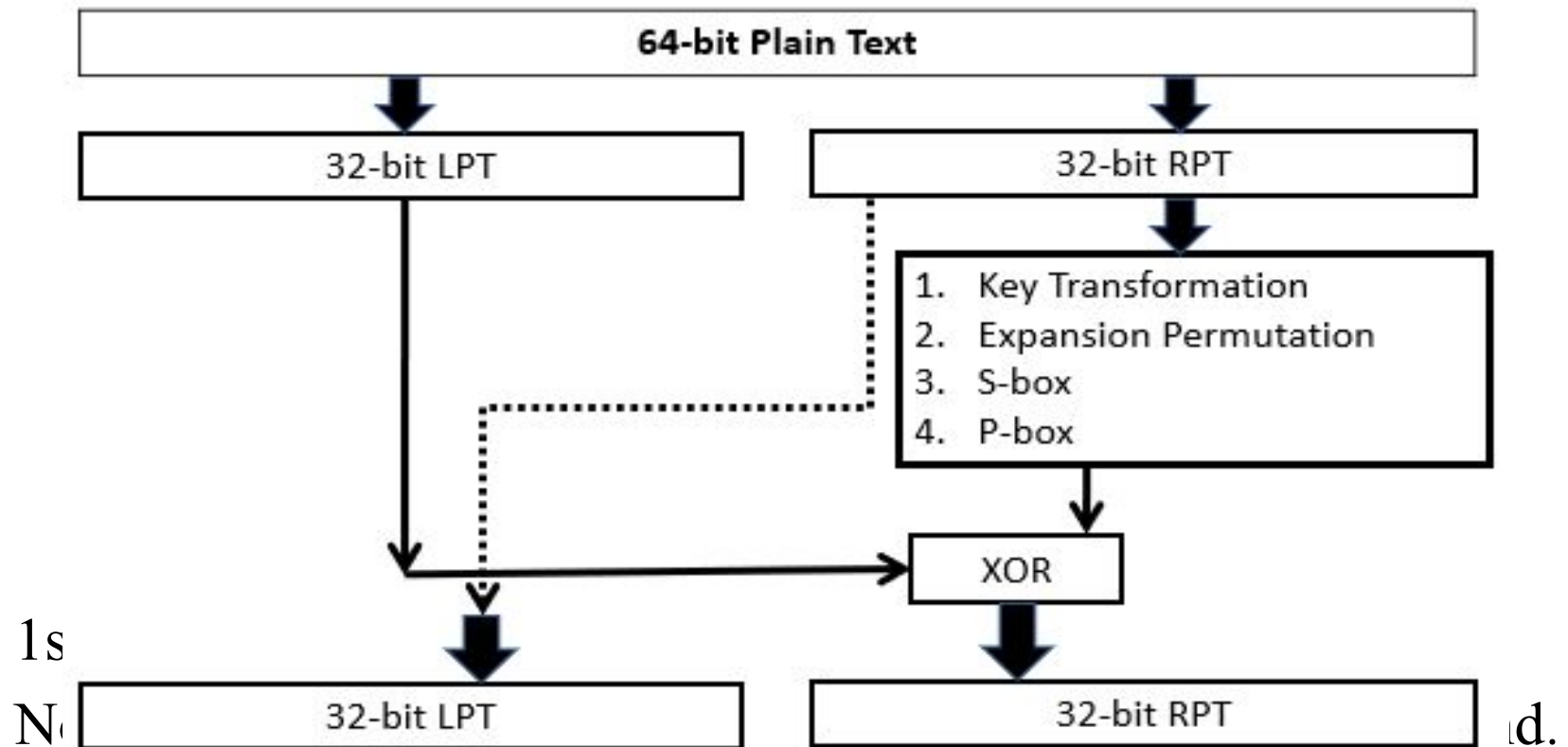
32-bit is permuted with 16 x 2 permutation table

For Example: 16th bit of S-box takes 1st Position as per below permutation table.

P – Box Table															
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

Step – 5: XOR and SWAP

32-bit LPT is XORed with 32-bit p-box.



Final Permutation

At the end of the 16 rounds, the final permutation is performed (only once).

For Example: 40th bit of input takes 1st Position as per below permutation table.

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

The output of the final permutation is the ***64-bit encrypted block (64-bit cipher text block)***.

ASYMMETRIC-KEY

• -

• Topics discussed in this section:

• RSA

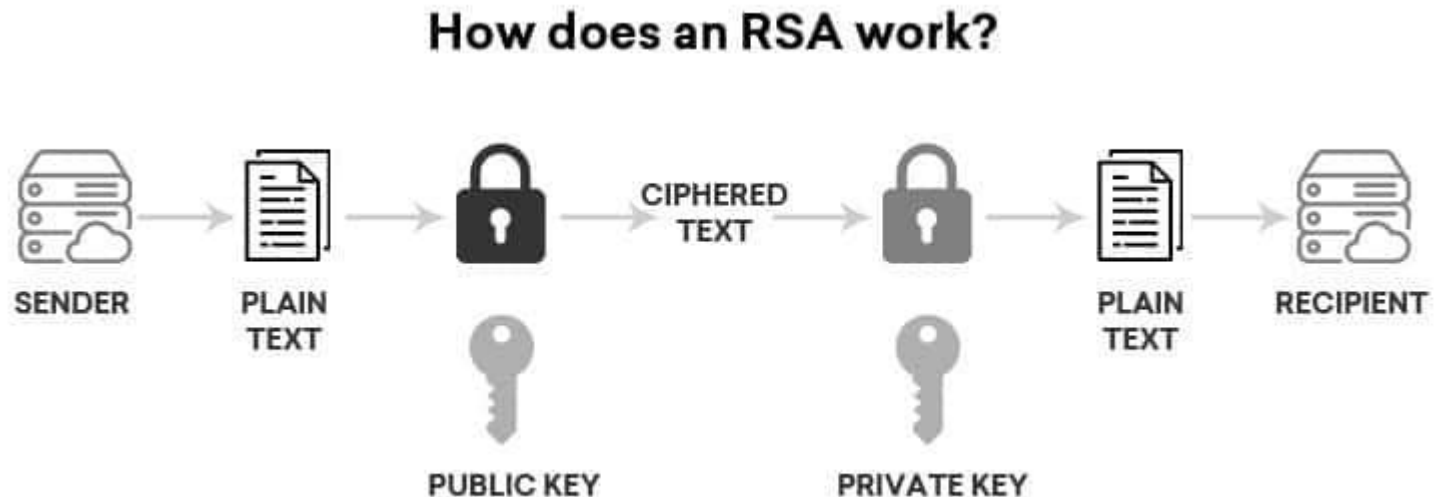
• Diffie-Hellman

Asymmetric encryption

- Asymmetric encryption, also known as public-key cryptography, uses a pair of keys (a public key and a private key) for encryption and decryption.
- Example - RSA(Rivest-Shamir-Adleman) Algorithm

RSA(Rivest-Shamir-Adleman) Algorithm

- It is an asymmetric or public-key cryptography algorithm which means it works on two different keys: Public Key and Private Key.
- RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.



RSA Algorithm

RSA Algorithm is based on factorization of large number and modular arithmetic for encrypting and decrypting data. It consists of three main stages:

- **Key Generation:** Creating Public and Private Keys
- **Encryption:** Sender encrypts the data using Public Key to get cipher text.
- **Decryption:** Decrypting the cipher text using Private Key to get the original data.

RSA Algorithm Steps

Step-1: Select two prime numbers p and q where $p \neq q$.

Step-2: Calculate $n = p * q$.

Step-3: Calculate $\Phi(n) = (p-1) * (q-1)$.

Step-4: Select e such that, e is relatively prime to $\Phi(n)$, i.e. $(e, \Phi(n)) = 1$ and $1 < e < \Phi(n)$

Step-5: Calculate $d = e^{-1} \bmod \Phi(n)$ or $ed = 1 \bmod \Phi(n)$.

Step-6: Public key = $\{e, n\}$, private key = $\{d, n\}$.

Step-7: Find out cipher text using the formula, $C = P^e \bmod n$
where, $P < n$ where C = Cipher text, P = Plain text, e = Encryption key and n =block size.

Step-8: $P = C^d \bmod n$. Plain text P can be obtain using the given formula.
where, d = decryption key

Example step by step

Step – 1: Select two prime numbers p and q where $p \neq q$.

Example, Two prime numbers $p = 13$, $q = 11$.

Step – 2: Calculate $n = p * q$.

Example, $n = p * q = 13 * 11 = 143$.

Step – 3: Calculate $\Phi(n) = (p-1) * (q-1)$.

Example, $\Phi(n) = (13 - 1) * (11 - 1) = 12 * 10 = 120$.

Step – 4: Select e such that, e is relatively prime to $\Phi(n)$,
i.e. $(e, \Phi(n)) = 1$ and $1 < e < \Phi(n)$.

Example, Select $e = 13$, $\gcd(13, 120) = 1$.

Step – 5: Calculate $d = e^{-1} \bmod \Phi(n)$ or $e * d = 1 \bmod \Phi(n)$

Example, Finding d: $e * d \bmod \Phi(n) = 1$

$$13 * d \bmod 120 = 1$$

(How to find: $d * e = 1 \bmod \Phi(n)$)

$$d = ((\Phi(n) * i) + 1) / e$$

$$d = (120 + 1) / 13 = 9.30 (\because i = 1)$$

$$d = (240 + 1) / 13 = 18.53 (\because i = 2)$$

$$d = (360 + 1) / 13 = 27.76 (\because i = 3)$$

$$d = (480 + 1) / 13 = 37 (\because i = 4)$$

Step – 6: Public key = $\{e, n\}$, private key = $\{d, n\}$.

Example, Public key = $\{13, 143\}$ and private key = $\{37, 143\}$.

Step – 7: Find out *cipher text* using the formula,

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

where, $P < n$.

Example, Plain text $P = 13$. (Where, $P < n$)

$$C = P^e \bmod n = 13^{13} \bmod 143 = 52.$$

Step – 8: $P = C^d \bmod n$.

Plain text P can be obtain using the given formula.

Example,

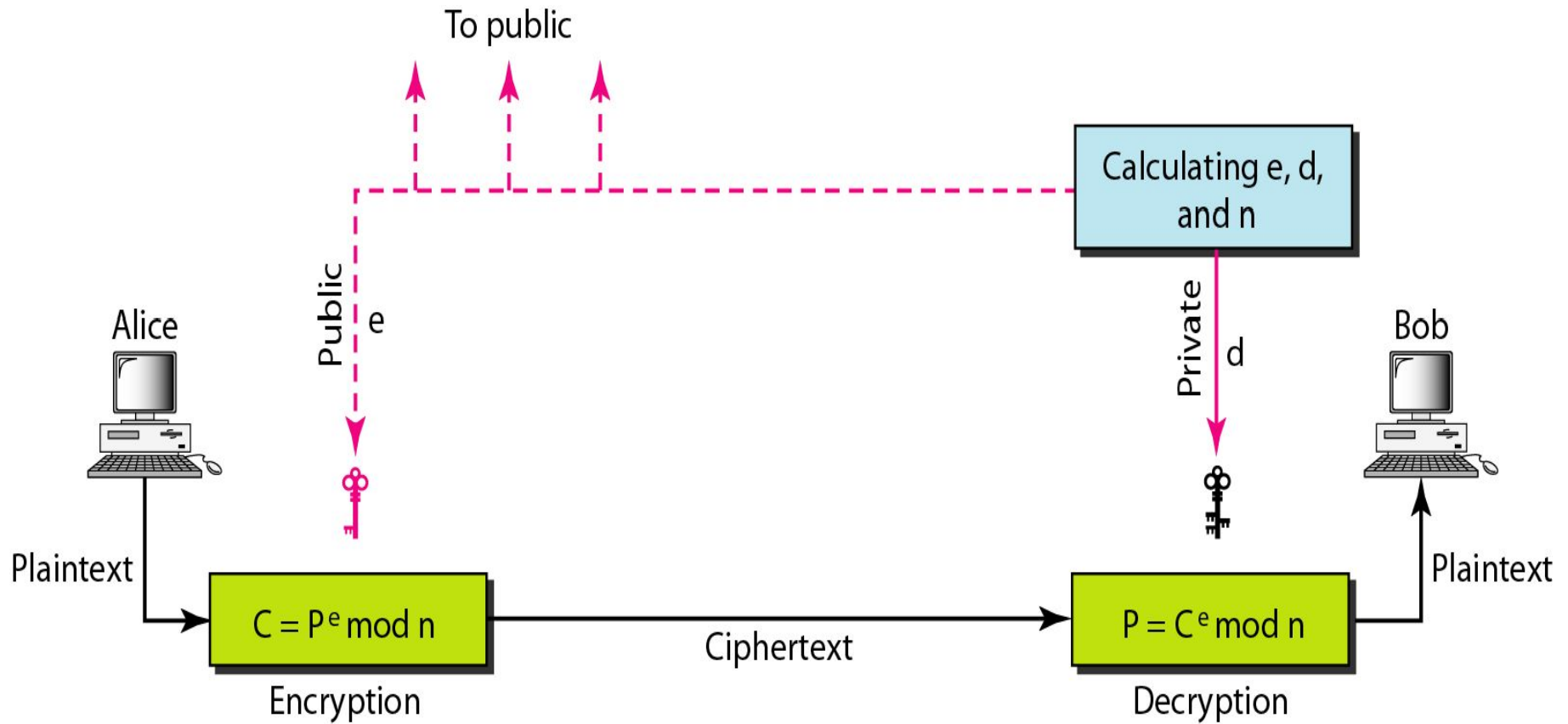
Cipher text $C = 52$

$$P = C^d \bmod n = 52^{37} \bmod 143 = 13.$$

Example

- $p = 61$ and $q = 53$ $e = 17$
- public key $(3233, 17)$ and the private key $(3233, 2753)$ text = HELLO
- **Encryption-**
- Suppose we want to encrypt the message "HELLO". We convert it to ASCII: H(72) E(69) L(76) L(76) O(79).
- We will encrypt each ASCII value separately –
- H: $C = 72^{17} \bmod 3233 = 2103$
- E: $C = 69^{17} \bmod 3233 = 2464$
- L: $C = 76^{17} \bmod 3233 = 2190$
- L: $C = 76^{17} \bmod 3233 = 2190$
- O: $C = 79^{17} \bmod 3233 = 875$
- **Decryption**
- The recipient receives the ciphertext $(2103, 2464, 2190, 2190, 875)$.
- They use their private key to decrypt each value –
- $M = 2103^{2753} \bmod 3233 = 72$ (H)
- $M = 2464^{2753} \bmod 3233 = 69$ (E)
- $M = 2190^{2753} \bmod 3233 = 76$ (L)
- $M = 2190^{2753} \bmod 3233 = 76$ (L)
- $M = 875^{2753} \bmod 3233 = 79$ (O)
- So, the decrypted message is "HELLO".

RSA





Note

In RSA, ***e*** and ***n*** are announced to the public; ***d*** and **Φ** are kept secret.

Example

Bob chooses 7 and 11 as p and q and calculates $n = 7 \cdot 11 = 77$. The value of $\Phi = (7 - 1) (11 - 1)$ or 60. Now he chooses two keys, e and d . If he chooses e to be 13, then d is 37. Now imagine Alice sends the plaintext 5 to Bob. She uses the public key 13 to encrypt 5.

Plaintext: 5

$$C = 5^{13} = 26 \bmod 77$$

Ciphertext: 26

Example

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26
 $P = 26^{37} = 5 \bmod 77$
Plaintext: 5

*The plaintext **5** sent by Alice is received as plaintext **5** by Bob.*

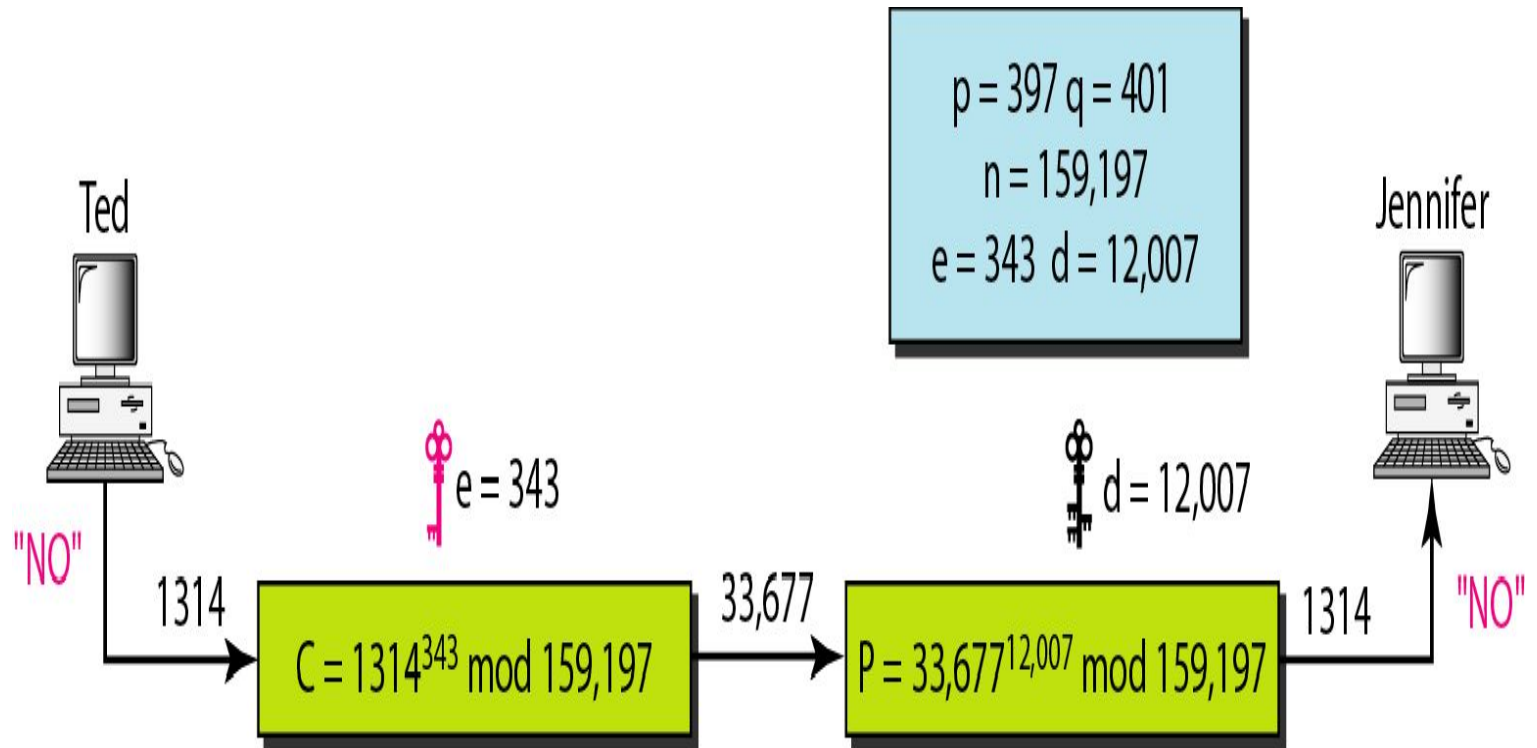
Example

Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159,197$ and $\Phi = 396 \cdot 400 = 158,400$. She then chooses $e = 343$ and $d = 12,007$. Show how Ted can send a message to Jennifer if he knows e and n .

Example (continued)

Solution

Suppose Ted wants to send the message “NO” to Jennifer. He changes each character to a number (from 00 to 25) with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Ted then uses e and n to encrypt the message. The ciphertext is $1314^{343} = 33,677 \bmod 159,197$. Jennifer receives the message 33,677 and uses the decryption key d to decipher it as $33,677^{12,007} = 1314 \bmod 159,197$. Jennifer then decodes 1314 as the message “NO”. Figure 30.25 shows the process.



Let us give a realistic example. We randomly chose an integer of 512 bits. The integer p is a 159-digit

$p =$ 96130345313583504574191581280615427909309845594996215822583150879647940
45505647063849125716018034750312098666606492420191808780667421096063354
219926661209

The integer q is a 160-digit number.

$q =$ 12060191957231446918276794204450896001555925054637033936061798321731482
14848376465921538945320917522527322683010712069560460251388714552496900
0359660045617

Example

We calculate n . It has 309 digits:

$n =$ 11593504173967614968892509864615887523771457375454144775485526137614788
54083263508172768788159683251684688493006254857641112501624145523391829
27162507656772727460097082714127730434960500556347274566628060099924037
10299142447229221577279853172703383938133469268413732762200096667667183
1831088373420823444370953

We calculate Φ . It has 309 digits:

$\phi =$ 11593504173967614968892509864615887523771457375454144775485526137614788
54083263508172768788159683251684688493006254857641112501624145523391829
27162507656751054233608492916752034482627988117554787657013923444405716
98958172819609822636107546721186461217135910735864061400888517026537727
7264467341066243857664128

Example 30.9

We choose $e = 35,535$. We then find d .

e = 35535

d = 58008302860037763936093661289677917594669062089650962180422866111380593852
82235873170628691003002171085904433840217072986908760061153062025249598844
48047568240966247081485817130463240644077704833134010850947385295645071936
77406119732655742423721761767462077637164207600337085333288532144708859551
36670294831

Alice wants to send the message “THIS IS A TEST” which can be changed to a numeric value by using the 00–26 encoding scheme (26 is the space character).

P = 1907081826081826002619041819

Example (continued)

The ciphertext calculated by Alice is $C = P^e$, which is.

$C = 4753091236462268272063655506105451809423717960704917165232392430544529$
6061319932856661784341835911415119741125200568297979457173603610127821
8847892741566090480023507190715277185914975188465888632101148354103361
6578984679683867637337657774656250792805211481418440481418443081277305
9004692874248559166462108656

Bob can recover the plaintext from the ciphertext by using $P = C^d$, which is

$P = 1907081826081826002619041819$

*The recovered plaintext is **THIS IS A TEST** after decoding.*



Note

**The symmetric (shared) key in
the Diffie-Hellman protocol is
 $K = g^{xy} \bmod p$.**

The Diffie-Hellman algorithm

- The Diffie-Hellman algorithm is a method for securely exchanging cryptographic keys over an insecure channel.
- The Diffie Hellman algorithm widely known as Key exchange algorithm or key agreement algorithm developed by Whitefield Diffie and Martin Hellman in 1976.

Step-1: Select q (prime number) and α (α is primitive root of q)

Step-2: User A Key Generation: select $X_A, X_A < q$

Calculate **public key** $Y_A, Y_A = \alpha^{X_A} \bmod q$

Y_A Shared with user B

Step-3: User B Key Generation: select $X_B, X_B < q$

Calculate **public key** $Y_B, Y_B = \alpha^{X_B} \bmod q$

Y_B shared with user A

Step-4: Calculation of **secret key by user A:** $K = (Y_B)^{X_A} \bmod q$

Step-5: Calculation of **secret key by user B:** $K = (Y_A)^{X_B} \bmod q$

Diffie – Hellman Key Exchange Algorithm explain with example:

Step – 1: Select q (prime number) and α (α is primitive root of q)

Example, here $q = 7, \alpha = 17$.

Step – 2: User A Key Generation: Select $X_A < q$,

calculate **public key** Y_A and shared with user B: $Y_A = \alpha^{X_A} \bmod q$

Example, Here $X_A = 6$,

Calculate $Y_A = \alpha^{X_A} \bmod q \Rightarrow 17^6 \bmod 7 \Rightarrow 1$

Step – 3: User B Key Generation: Select $X_B < q$,

calculate **public key** Y_B and shared with user A: $Y_B = \alpha^{X_B} \bmod q$

Example, Here $X_B = 4$,

Calculate $Y_B = \alpha^{X_B} \bmod q \Rightarrow 17^4 \bmod 7 \Rightarrow 4$

Step – 4: Calculation of **secret key** by user A:

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

Example, Here $Y_B = 4, X_A = 6$

Calculate $K = (Y_B)^{X_A} \bmod q \Rightarrow 4^6 \bmod 7 \Rightarrow 1$

Step – 5: Calculation of **secret key by user B:**

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

Example, Here $Y_A = 1, X_B = 4$

Calculate $K = (Y_A)^{X_B} \bmod q \Rightarrow 1^4 \bmod 7 \Rightarrow 1$

Example

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume $g = 7$ and $p = 23$. The steps are as follows:

- 1. Alice chooses $x = 3$ and calculates $R_1 = 7^3 \bmod 23 = 21$.*
- 2. Bob chooses $y = 6$ and calculates $R_2 = 7^6 \bmod 23 = 4$.*
- 3. Alice sends the number 21 to Bob.*
- 4. Bob sends the number 4 to Alice.*
- 5. Alice calculates the symmetric key $K = 4^3 \bmod 23 = 18$.*
- 6. Bob calculates the symmetric key $K = 21^6 \bmod 23 = 18$.*

The value of K is the same for both Alice and Bob; $g^{xy} \bmod p = 7^{18} \bmod 23 = 18$.

Figure 30.27 *Diffie-Hellman idea*

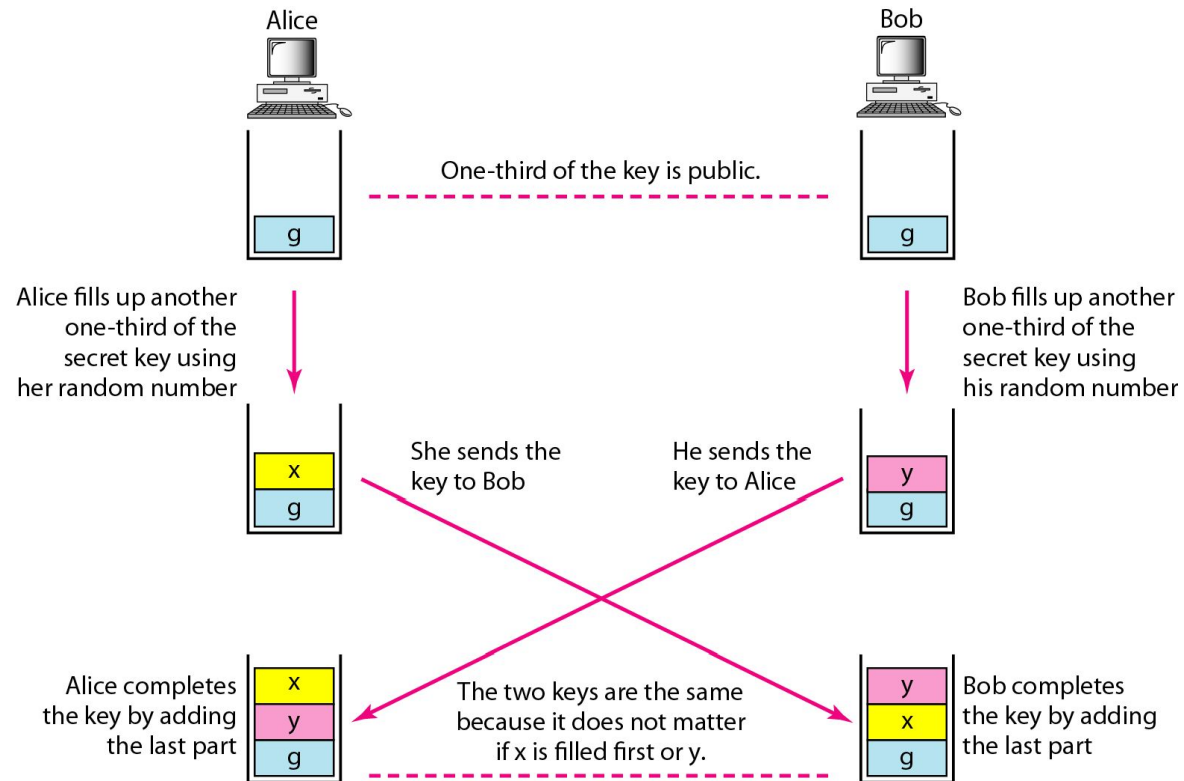
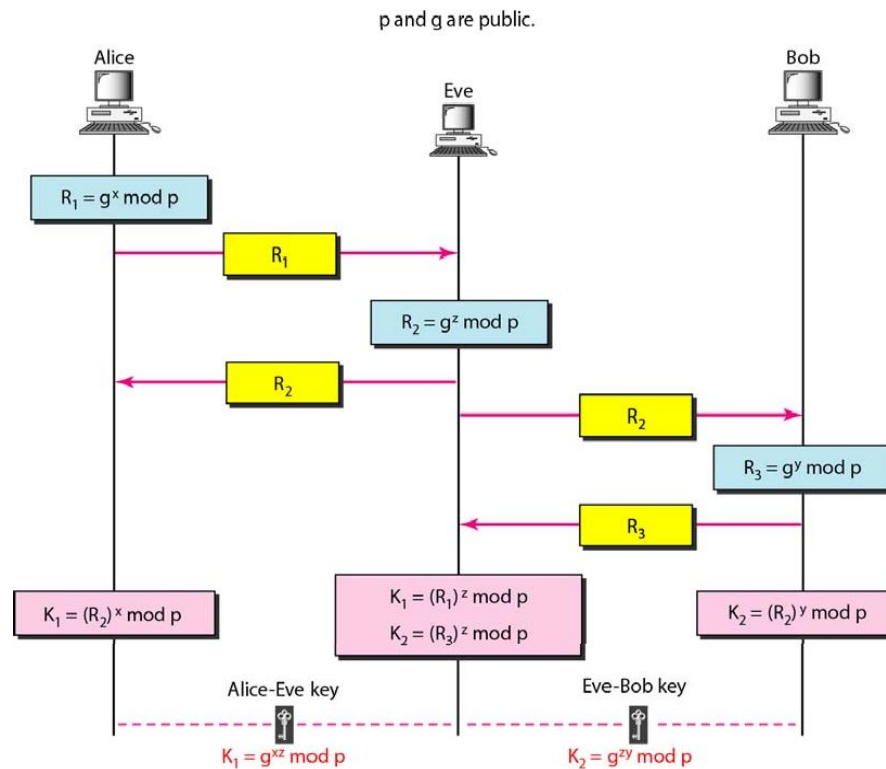


Figure 30.28 *Man-in-the-middle attack*



Digital Signature

- A digital signature is a cryptographic technique used to verify the authenticity, integrity, and non-repudiation of digital data or messages. It is widely used in various digital transactions, communications, and document management systems.

Uses of Digital Signatures

1. **Authentication:** Confirms the identity of the sender or signer. Ensures the message or document is from a verified source (e.g., verifying the identity of someone signing a contract online).
2. **Data Integrity:** Detects any changes made to a document or message after it was signed. If data is altered, the signature becomes invalid.
3. **Non-repudiation:** Prevents the signer from denying that they signed a document. Legally binds the signer to the contents.
4. **Secure Email Communication:** Digitally signed emails verify the sender and ensure the email hasn't been tampered with.
5. **Software Distribution:** Developers use digital signatures to prove that software hasn't been altered after release. Helps users trust downloaded programs (e.g., Windows executable files often have publisher signatures).
6. **E-Governance and E-Transactions:** Used in e-filing of taxes, business registrations, and online applications (e.g., in India's Digital India initiative). Ensures secure and legally valid digital documentation.
7. **Blockchain and Cryptocurrencies:** Ensures transaction authenticity and integrity in cryptocurrencies like Bitcoin and Ethereum.
8. **Digital Contracts and Legal Documents :** Used in digital contracts to give them the same legal standing as handwritten signatures.

Hashing

Introduction

- In hash function H accepts a variable length block of input data called as 'M' and produces the fixed size hash value can be represented as $h = M(H)$.

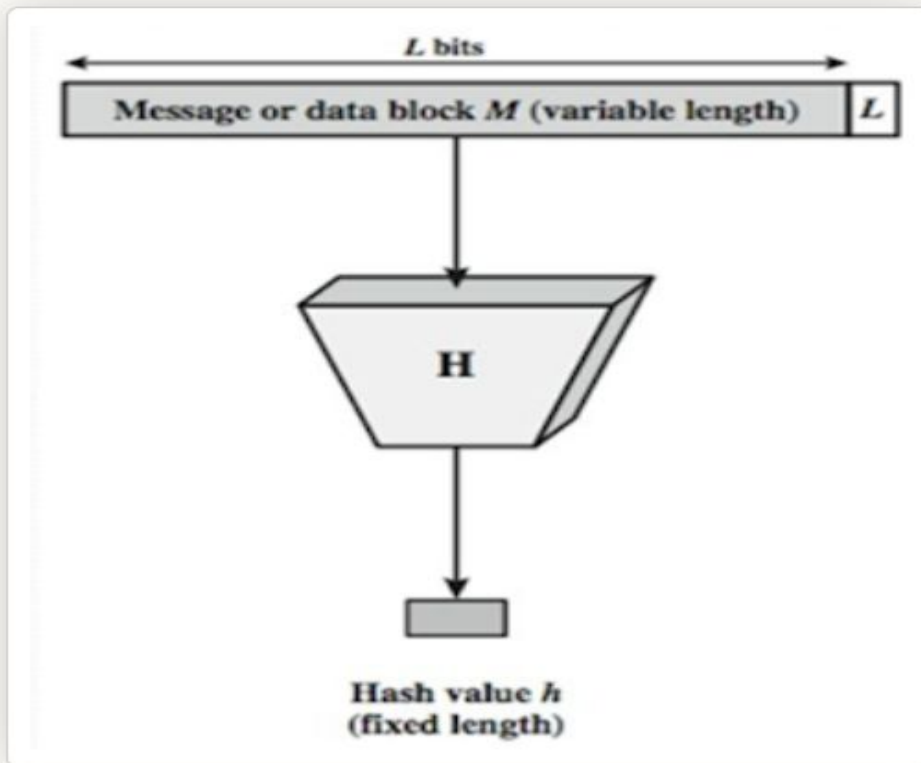
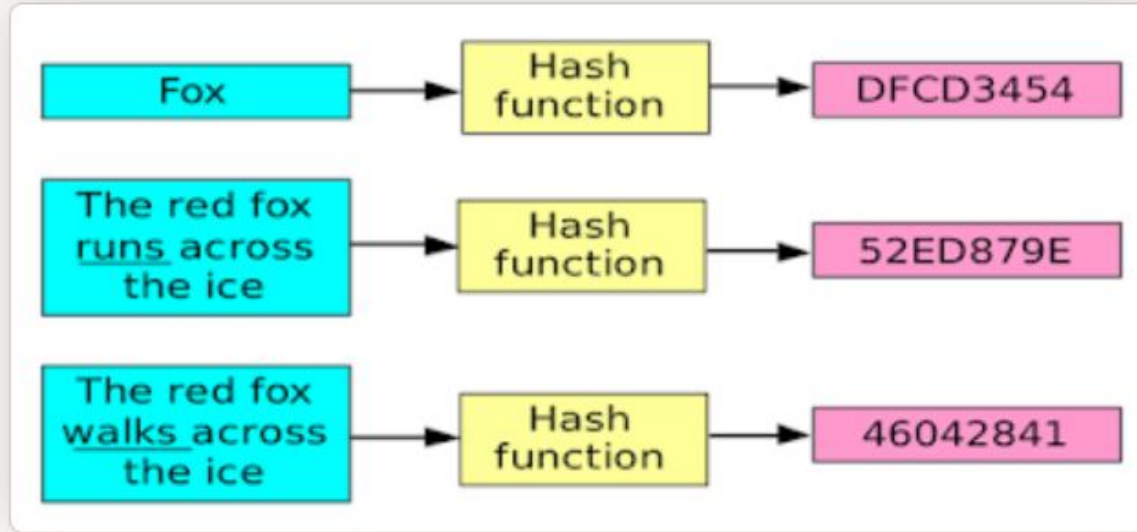


Figure: Block Diagram of hash function

When hash function provides security, this is called cryptographic hash functions. Hash function protects the integrity of the message. If encryption process is applying on message with hash function, it is also providing authentication and confidentiality.



A hash function provides a property that has function applied on variable amount of data (M) and then it produces the fixed amount of output data.

If any bit or bits changes in the data, then whole hash function output data will also change.

Cryptographic hash function is one-way function, which is practically infeasible to invert.

The most popular hashing algorithm is MD5 and SHA.

Properties of hash Function

- Compression: As per compression properties, output of the hash function is much smaller than the size of input.
- Pre-image resistance: Pre-image resistance means difficult to find the input from given hash function output. i.e., $x=H(m)$. So if x is given, it is difficult to message m .
- Weak Collision Resistance: Given message m_1 , weak collision resistance means that it is difficult to produce another message m_2 such that $H(m_1)=H(m_2)$. i.e, it means it is infeasible to find two different messages with the same hash value.
- Strong Collision Resistance: Strong collision resistance means that is difficult to find any two different messages that hash to the same value. i.e., it means it is hard to find m_1 & m_2 such that same hash value $H(m_1) = H(m_2)$.
-

Characteristics of hash Function

- 1. It is quick to calculate hash value (h) for any given message. i.e., $x = H(m)$.
- 2. Hash function (H) can be applied to variable length of data block.
- 3. A small change in a message should change the hash value.
- 4. Hash function has one-way property; it is impossible to generate message from given hash value.
- 5. The hash function uses all the input data.
- 6. The hash function "uniformly" distributes the data across the entire set of possible hash values.
- 7. The hash function generates very different hash values for similar message.

Message Digest 5 algorithm

- 128-bit message digest developed by Ron Rivest. This algorithm takes the input length of arbitrary length and 128-bit message digest is produced. The input message is 512-bit blocks. Figure shows processing of message to produce message digest.

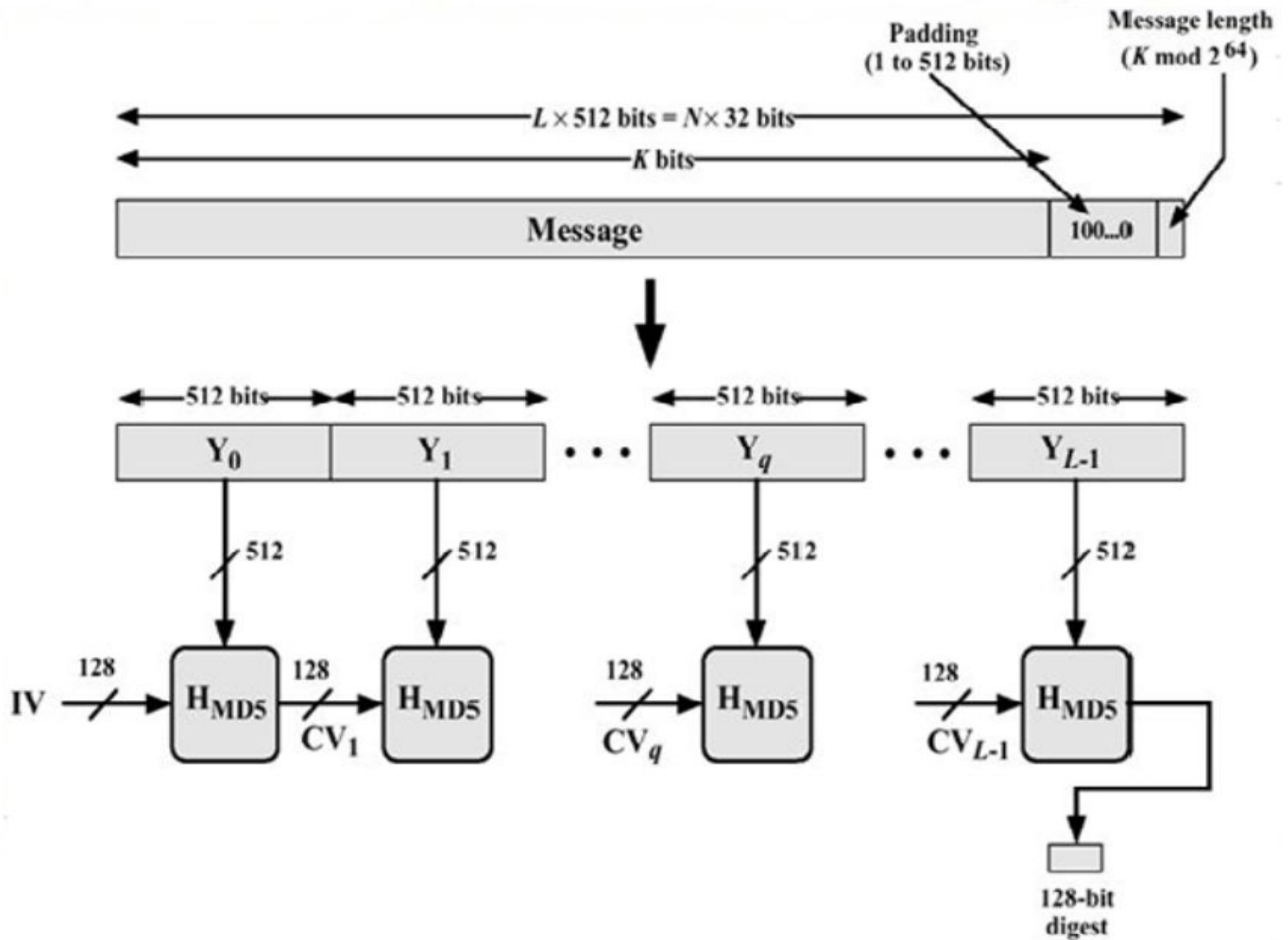


Figure: Message Digest generation using MD5

MD 5 Algorithm

- Step 1: Append Padding Bits – The message is padded to make the length of message is $448 \bmod 512$. 64 bits is padded with 448 bits and convert into multiple of 512 bits. The padding message consists a single 1-bit followed by 0 bits. The length of padding bits is in between 1 to 512.
- Step 2: Append Length – 64 bit of original message is appended to the result of above step 1. It is appended such that least significant bytes to most significant byte. The output of step 2 yields a message of integer multiple of 512 bits. As $M_0, M_1, \dots, M_q, \dots, M_{L-1}$. The total length of expended message is $L * 512$ bits.

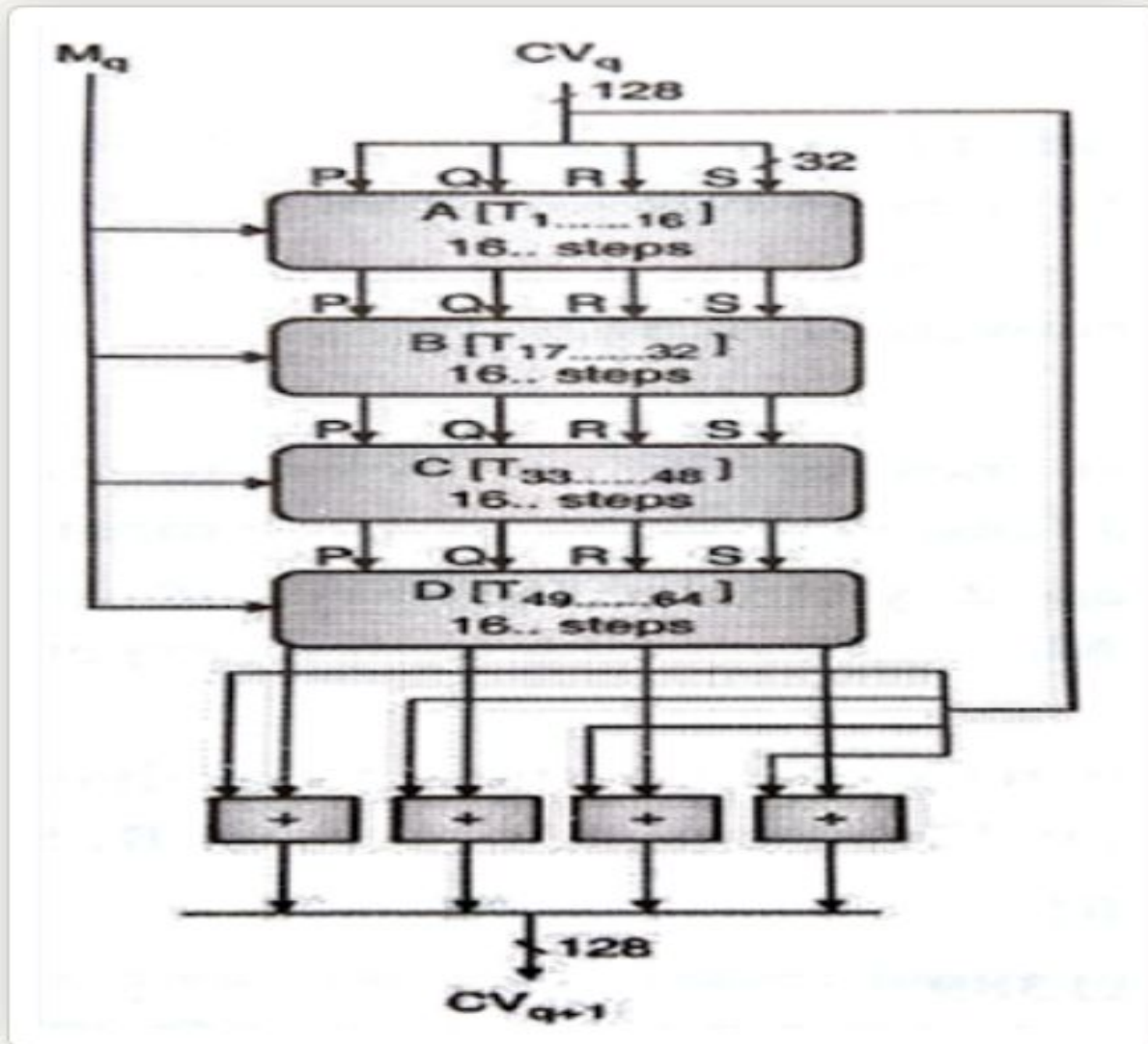


Figure: Intialized MD Buffer

- Step 3: Initialize MD Buffer – A 128-bit buffer is used to store the intermediate as well as final result. A buffer is represented as four 32-bit registers as four 32-bit registers as P, Q, R, S.

P = 01 23 45 67

Q = 89 AB CD EF

R = FE DC BA 98

S = 76 54 32 10

- It is used an initial value (IV).
- Step 4: Process Message in 512-bit blocks – It consists of four rounds of processing as shown in figure. These four rounds have similar structure as SHA but differ in primitive logical function referred as A, B, C, D. Each round takes input 512-bit block, processed it and produces 128-bit output. The output of fourth round is added to the first round CVq to produce CVq+1.
- Step 5: Output – After processing all L 512-bit blocks, the 128-bit message digest is produced as an output.

Introduction

- The secure hash algorithm (SHA) was developed by National Institute of Standards and Technology (NIST). It is based on MD4 algorithm. Based on different digest lengths, SHA includes algorithms such as SHA-1, SHA-256, SHA-384 and SHA-512. Unlike encryption, given a variable length message x , a secure hash algorithm computes a function $H(x)$ which has a fixed bit.
- When a message of any length is less than 264 bits is input, the SHA-1 produces a 160-bit output called message digest. SHA-1 called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.
- The most commonly used hash function from the SHA family is SHA-1. SHA-1 is used in SSL/TLS, PGP, SSH, MIME and IPsec for security and authentication purpose.
-

Features of SHA – 1

- Message or data file used as input in SHA-1 to compute a message digest (output of hash function or final hash value).
- The message or data file should be considered to be a bit string. The length of the message is the number of bits in the message (the empty message has length 0).
- The purpose of message padding is to make the total length of a padded message a multiple of 512. (If any message length is 1000 bits, so padded 24 bits to make message into multiple of 512 bits). The SHA-1 sequentially processes blocks of 512 bits when computing the message digest.

Working of SHA – 1

- SHA1 works with any input message that is less than 264 bits in length. The output of SHA is a message digest, which is 160 bits in length

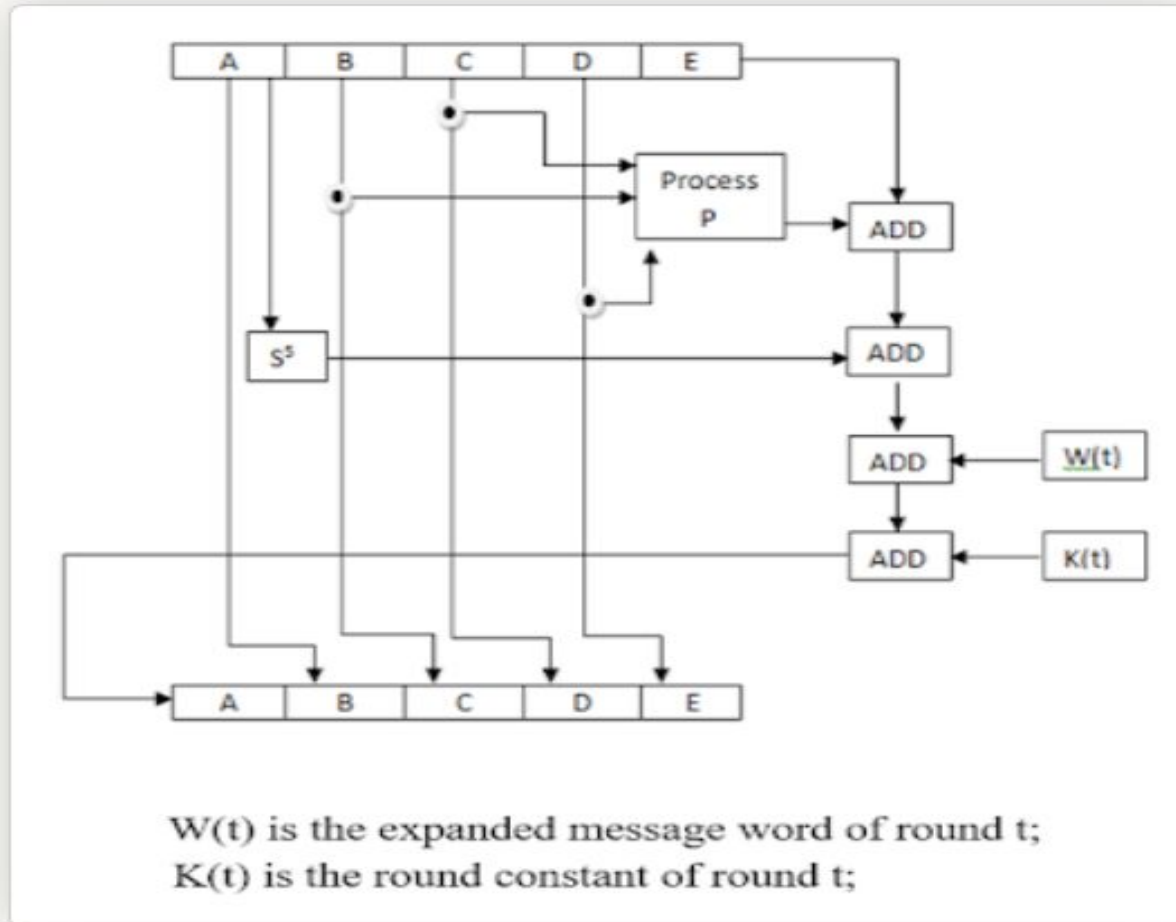


Figure: Working of SHA-1

- Step – 1: Padding - The first step of SHA-1 is added padding to the end of original message to prepare message in multiple of 512 bits.
- Step – 2: Append Length – The length of message excluding the length of the padding is now calculated and appended to the end of the padding as 64-bit block. (message length is 64 bits short of multiple of 512).
- Step – 3: Divide the input into 512-bit blocks: The input message is now divided into blocks, each of length 512 bits.
- Step – 4: Initialize chaining variables: Now, five chaining variables A to E are initialized. Each of 32 bits variable produces 160 bits length of message digest.
- Step – 5: Process Block & Output – Combination of A-E chaining variable is called ABCDE, will be considered as a single register. Now divided the current 512-bit block into 16 sub blocks, each consisting of 32 bits. (32x16=512) SHA-1 has perform four rounds. Each round takes the current 512-bit block, the register ABCDE and constant K(t) (where t=0 to 79) as input. SHA consists of four rounds, each round containing 20 iterations. So total iteration is 80. The logical operation of a single SHA-1 iteration looks as shown in figure. Mathematically, an iteration consists of the following operation:
 - $ABCDE = E + \text{Process } P + S 5(a) + W(t) + K(t)$

Sr. No.	Parameters	SHA-1	SHA-256	SHA-384	SHA-512
1	Message digest size	160	256	384	512
2	Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
3	Block size	512	512	1024	1024
4	Word size	32	32	64	64
5	Number of steps	80	64	80	80
6	Security level	80	128	192	256

Note: All values are measured in bits.

Difference between SHA-1 and MD 5

No	SHA - 1	MD 5
1	It uses a 160-bit message digest.	It uses a 128-bit message digest
2	It is stronger against Brute-force attack compare to MD5.	It is weak against Brute-force attack compare to SHA-1.
3	SHA - 1 is not vulnerable against cryptanalysis.	MD5 is vulnerable against cryptanalysis.
4	SHA - 1 is slower than MD5.	MD5 is faster than SHA-1.
5	It uses big-endian method to represent the message.	It uses little-endian method to represent the message.
6	SHA has 80 steps	MD5 has 64 steps.