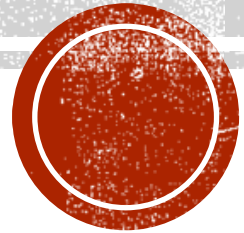
CHAPTER 2: SYMMETRIC AND ASYMMETRIC CRYPTOGRAPHICS ALGORITHMS



LH- 13

ROLISHA STHAPIT

CONTENTS

- Classical Cryptosystems: Substitution and Transposition Ciphers, Block Cipher Vs Stream Ciphers, Symmetric Encryption Principles, Fiestel Cipher Structure, Data Encryption Standards (DES), Basic concepts of fields, Modular Arithmetic, Galois Fields, Polynomial Arithmetic, Advanced Encryption Standards (AES), Prime Numbers, Fermat's Theorem, Primality Testing: Miller-Rabin Algorithm, Euclidean Algorithm, Extended Euclidean Algorithm, Euler Totient Function, Asymmetric Encryption, Diffie Hellman Protocol, RSA Algorithm

- **What is Cryptography?**
- Cryptography is 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.
- In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher (decode). These deterministic algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet and confidential communications such as credit card transactions and email.
- Cryptography is closely related to the disciplines of cryptology and cryptanalysis. It includes techniques such as microdots, merging words with images and other ways to hide information in storage or transit. However, in today's computer centric world, cryptography is most often associated with scrambling (mix up) plaintext (ordinary text, sometimes referred to as cleartext) into ciphertext (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers.

- **What is cryptology?**
- Cryptology is the study of mathematics, codes and equations, such as number theory and the application of formulas and algorithms that supports cryptography and cryptanalysis.
- **What is Cryptanalysis?**
- Cryptanalysis is the practice of analyzing cryptographic systems and algorithms in order to find flaws (fault) and vulnerabilities to decrypt the cipher-text by unauthorized user.
- For example, cryptanalysts attempt to decrypt ciphertexts without knowledge of the encryption key or algorithm used for encryption. Cryptanalysts use their research results to help to improve and strengthen or replace flawed algorithms.

- **Cryptosystems:**
- A cryptosystem is a structure or scheme consisting of a set of algorithms that converts plaintext to ciphertext to encode or decode messages securely. The term “cryptosystem” is shorthand for “cryptographic system” and refers to a computer system that employs cryptography, 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.
- To help keep data secure, cryptosystems incorporate the algorithms for key generation, encryption and decryption techniques. At the heart of cryptographic operations is a cryptographic key, a string of bits used by a cryptographic algorithm to transform plain text into ciphertext or the reverse. The key is part of the variable data provided as input to a cryptographic algorithm to execute this sort of operation. The cryptographic scheme’s security depends on the security of the keys used.
- Cryptosystems are used for sending messages in a secure manner over the internet, such as credit card information and other private data. In another application of cryptography, a system for secure electronic mail might include methods for digital signatures, cryptographic hash functions and key management techniques.

- **Components of Cryptosystems:**
- A basic cryptosystem includes the following components:
- **Plaintext:** This is the data that needs to be protected.
- **Encryption algorithm:** This is the mathematical algorithm that takes plaintext as the input and returns ciphertext. It also produces the unique encryption key for that text.
- **Ciphertext:** This is the encrypted, or unreadable, version of the plaintext.
- **Decryption algorithm:** This is the mathematical algorithm that takes ciphertext as the input and decodes it into plaintext. It also uses the unique decryption key for that text.
- **Encryption key**: This is the value known to the sender that is used to compute the ciphertext for the given plaintext.
- **Decryption key:** This is the value known to the receiver that is used to decode the given ciphertext into plaintext.

- **Types of Cryptosystems:**
- Cryptosystems are categorized by the method they use to encrypt data, either symmetrically or asymmetrically.

1. **Symmetric key encryption:** It is when the cryptosystem uses the same key for both encryption and decryption. In this method, keys are shared with both parties prior to transmission and are changed regularly to prevent any system attacks.

Fig: Symmetric Key Encryption

2. Asymmetric key encryption:

- It is when the cryptosystem uses different keys for encryption and decryption. However, the keys are mathematically related. In this method, each party has their own pair of keys (private key and public key) that is exchanged during transmission. Message encrypted by public key can decrypt using corresponding private key and vice-versa.

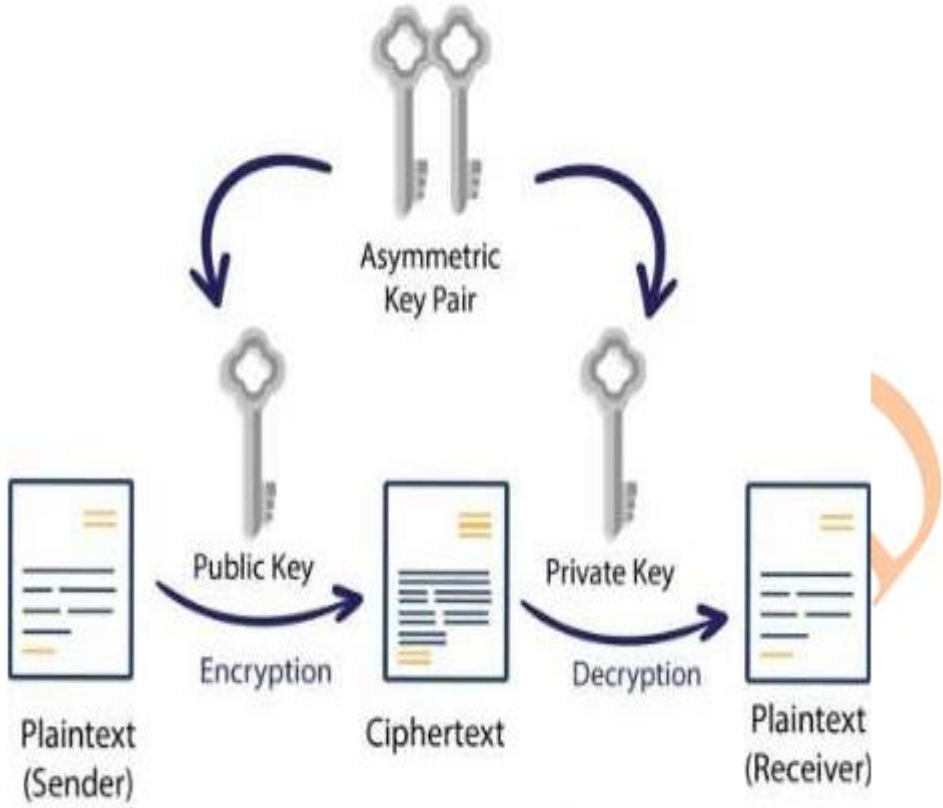


Fig: Asymmetric Key Encryption

Classical Cipher

- The historic technique that used only the pen, pencil and normal mathematics to hide information.
- Classical ciphers are too weak to use now and can be broken easily with computer.
- Use the single key for both encryption and decryption.
- Two basic components of classical ciphers: ***Substitution*** and ***Transposition.***

- ***Substitution Cipher:*** A substitution is a technique in which each letter or bit of plaintext is substituted or replaced by some other letter, number or symbol to produce cipher text. E.g. Caesar, Playfair, Hill Cipher etc.

- ***Transposition Cipher:*** In transposition technique, there is no replacement of alphabets or numbers occurs instead their positions are changed or reordering of position of plaintext is done to produce cipher text. E.g. Rail Fence Cipher

Substitution Cipher Types

- ***Monoalphabetic Cipher***
  - The cipher alphabet for each plain alphabet is fixed throughout the encryption process. E.g. If 'A' gets substituted by 'E' then every occurrence of 'A' will be substituted by 'E'.
  - In monoalphabetic cipher, the relationship between a character in the plaintext and the character in the ciphertext is one-to-one.

- ***Polyalphabetic cipher***
  - The cipher alphabet for the plain alphabet may be different at different places during the encryption process. E.g. Vigenere cipher
  - In polyalphabetic cipher, the relationship between a character in the plaintext and the character in the ciphertext is one-to-many.

CEASER CIPHER

- The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It‟s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 2, A would be replaced by C, B would become D, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

- Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down. The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

- Encryption Phase with shift n is written as:

- ***Cipher-text = E(p+n) mod 26***

- Decryption Phase with shift n is written as:

- ***Plain-text = D(c-n) mod 26***

Encryption of a letter by a shift k can be described mathematically as,

$$c = (m + k) \bmod 26$$

Similarly, Decryption

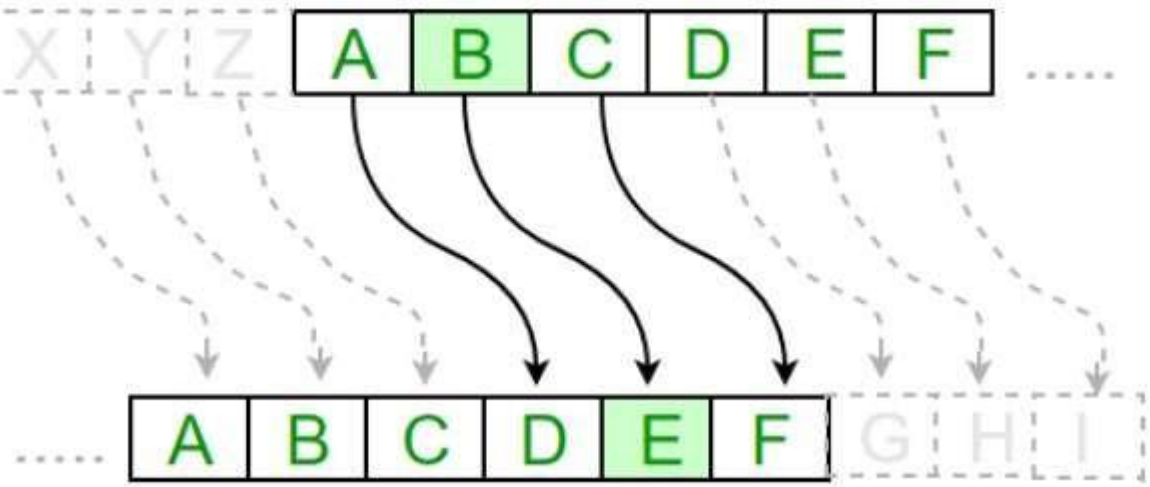
$$m = (c + 26 - k) \bmod 26$$

$$\begin{cases} m = Plaintext \\ c = Ciphertext \\ k = key \end{cases}$$

Here, we number each English alphabet starting from 0 (A) to 25 (Z).

Where,

E = Encryption
D = Decryption
n = shift the alphabets by n position p = alphabet of plain text
c = alphabet of cipher text



- Assign the number for all the alphabets as:

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

- **Example 1:** Encrypt “hello zenny” using Caesar Cipher.

- Use the encryption key as 3. **Solution:**
- Cipher-text = (p+3) mod 26
- Where, p is the letters of plain text.

PT	**H**	**E**	**L**	**L**	**O**	**Z**	**E**	**N**	**N**	**Y**
CT	K	H	O	O	R	C	H	Q	Q	B

- H = 7, so
- $E_3(h) = (7+3) \bmod 26 = 10$
- $E_3(z) = (25+3) \bmod 26 = 2$
- $E_3(y) = (24+3) \bmod 26 = 1$
- **Plain Text** = hellozenny
- **Cipher Text** = khoorchqqb

Example 2: Decrypt “khoorchqqb” using Caesar Cipher.

- Use the decryption key as 3.
- **Solution:**
- Plain-text = (c-3) mod 26
- Where, c is the letters of cipher-text.

CT	**K**	**H**	**O**	**O**	**R**	**C**	**H**	**Q**	**Q**	**B**
PT	H	E	L	L	O	Z	E	N	N	Y

K = 10, so

$D_3(k) = (10-3) \bmod 26 = 7$

$E_3(c) = (2-3) \bmod 26 = 2$

$= (-1) \bmod 26$ (when you get –ve in left of the mod then add to 26)

$= (-1+26) \bmod 26$

$= 25 \bmod 26$

$= 25$

$E_3(b) = (1-3) \bmod 26 = 24$

Cipher Text = khoorchqqb

Plain Text = hellozenny

ONE TIME PAD (VERNAM CIPHER)

- It is a type of substitution cipher.
- One time pad technique uses a random key of the same length of message.
- **Each key is used only once, and both sender and receiver must destroy** their key after use.
- There should be only two copies of the key: one for the sender and one for the receiver.
- Sender generates new key for every new message while sending message to receiver so it called as one time pad.

Encryption:

$$C = (P + K) mod\ 26$$

Decryption:

$$P = (C - K) mod\ 26$$

Q. Encrypt and decrypt the message "HELLO" with the key "NCBTA" using One time pad cipher.

Solⁿ:

Plaintext: HELLO
Key: NCBTA

Encryption:

Plaintext (P)	H	E	L	L	O
Numerical Plaintext	7	4	11	11	14
			+		
Key (K)	N	C	B	T	A
Numerical Key	13	2	1	19	0
P + K	20	6	12	30	14
(P+K) mod 26	20	6	12	4	14
Ciphertext(C)	V	G	M	E	O

$\therefore Ciphertext = VGMEO$

Decryption:

Ciphertext(C)	V	G	M	E	O
Numerical Ciphertext	20	6	12	4	14
			-		
Key (K)	N	C	B	T	A
Numerical Key	13	2	1	19	0
C - K	7	4	11	-15	14
(C-K) mod 26	7	4	11	11	14
Plaintext (P)	H	E	L	L	O

$\therefore Plaintext = HELLO$

VIGENERE CIPHER

- It is a **Polyalphabetic** substitution cipher.
- It is based on the matrix of alphabet i.e. 26 × 26 matrix.
- Plaintext is assumed to be in row and key is assumed to be in column.
- The length of key and plaintext must be same. (Repeat the letters of key over and over until it is the same length as the plaintext).

Encryption:

- Ciphertext letter is the intersection of plaintext letter and corresponding key letter in the table.

Decryption:

- Decryption is performed by finding the position of the ciphertext letter in a column, corresponding to the key letter, of the table, and then taking the label of the row in which it appears as the plaintext letter.

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:
- Encrypt the plain text “weareincwcqualifier” using vigenere cipher and also decrypt the result back to plain text. Use key as: thankyou
- Step 1: Make key length equals to the plain text by repeating the key value
- Plain text = weareincwcqualifier
- Key = thankyouthankyoutha
- Now the length of plain text and key is same.

PT	W	E	A	R	E	I	N	C	W	C	Q	U	A	L	I	F	I	E	R
Key	T	H	A	N	K	Y	O	U	T	H	A	N	K	Y	O	U	T	H	A
CT	P	L	A	E	O	G	B	W	P	J	Q	H	K	J	W	Z	B	L	R

Plain Text: weareincwcqualifier

Cipher Text: plaeogbwpjqhkjwzblr

- For decryption: Look at the row of key and find the cipher alphabet, then see the which column of plain text it belongs.

Key	T	H	A	N	K	Y	O	U	T	H	A	N	K	Y	O	U	T	H	A
CT	P	L	A	E	O	G	B	W	P	J	Q	H	K	J	W	Z	B	L	R
PT	W	E	A	R	E	I	N	C	W	C	Q	U	A	L	I	F	I	E	R

Cipher Text: plaeogbwpjqhkjwzblr
Plain Text: weareincwcqualifier

RAIL FENCE CIPHERS

- The Rail Fence cipher is an easy to apply transposition cipher that jumbles up the order of the letters of a message in a quick convenient way. It also has the security of a key to make it a little bit harder to break.
- The Rail Fence cipher works by writing your message on alternate lines across the page, and then reading off each line in turn
- The rail fence cipher (also called a zigzag cipher) is a form of transposition cipher. It derives its name from the way in which it is encoded.
- **Encryption Algorithm**: In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

1. In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence.
2. When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner.
3. After each alphabet has been written, the individual rows are combined to obtain the cipher-text.

- **Decryption Algorithm:** As we've seen earlier, the number of columns in rail fence cipher remains equal to the length of plain-text message. And the key corresponds to the number of rails.

1. Hence, rail matrix can be constructed accordingly. Once we've got the matrix we can figure-out the spots where texts should be placed (using the same way of moving diagonally up and down alternatively).
2. Then, we fill the cipher-text row wise. After filling it, we traverse the matrix in zig-zag manner to obtain the original text.

- Example: Encrypt the message “prime college” using rail fence algorithm with depth 3. And also decrypt it.

Solution:

Plain Text: prime college

Key: 3 Create a rail fence matrix No. of rows = 3 (depth of rail fence) No. of columns = 12 (length of plain text)

P				E				L			
	R		M		C		L		E		E
		I				O				G	

Cipher Text: pelrmcleeiog

Decryption:

Cipher Text: pelrmcleeiog

Key: 3

Create rail fence matrix

No. rows = 3 (depth/key)

No. of column = 12 (length of cipher text)

First, make a matrix and find the position of the cipher characters

*				*				*			
	*		*		*		*		*		*
		*				*				*	

Then, fill the cipher character row wise and read diagonally.

P				E				L			
	R		M		C		L		E		E
		I				O				G	

Plain text: primecollege

Q. Given the plaintext "ABRA KA DABRA", compute the ciphertext for the Railfence cipher with rails = 3.

Soln:

Plaintext: ABRA KA DABRA

No. of rails = 3

A				K				B		
	B		A		A		A		R	
		R				D				A

$\therefore Ciphertext = AKBBAAARRDA$

Q. Encrypt the plaintext "CAPTAIN AVENGER" with rails = 4 using Railfence cipher.

Soln:

Plaintext: CAPTAIN AVENGER

No. of rails = 4

C						N						E	
	A				I		A				G		R
		P		A				V		N			
			T						E				

$\therefore Ciphertext = CNEAIAGRPAVNTE$

Q. Decrypt the ciphertext "CAVEATIAEGRPNN" with no. of rails = 3 using Railfence cipher.

Sol^{n}:

Ciphertext: CAVEATIAEGRPNN

No. of rails = 3

*				*				*				*	
	*		*		*		*		*		*		*
		*				*				*			

C				A				V				E	
	A		T		I		A		E		G		R
		P				N				N			

∴Plaintext =CAPTAINAVENGER

PLAYFAIR CIPHER

- The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher.
- In playfair cipher we encrypt a pair of alphabets (digraphs) instead of a single alphabet. It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.
- **Algorithm:**
- Step 1: Generate the key Square (5×5):

- The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

- Step 2: Generate pairs of plain text:

- The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a filler character is added to the last letter. The filler character may be any alphabet suppose x.

- For example:

- PlainText: "instruments"

- After Split: 'in' 'st' 'ru' 'me' 'nt' 'sx’

- Pair cannot be made with same letter. Break the letter in single and add a filler character to the previous letter.

- For Example:

- Plain Text: “hello”

- After Split: he lx lo

- Here „x“ is the bogus letter.

- **Rules for Encryption:**

- If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom).
- If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).
- If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

- **Rules for Decryption:**

- If both the letters are in the same column: Take the letter above each one (going back to the bottom if at the top).
- If both the letters are in the same row: Take the letter to the left of each one (going back to the rightmost if at the leftmost position).
- If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.
- Remove filler character if available in the result

- Example: Encrypt the plain text “my name is jenney” using playfair cipher use the encryption key as “monarchya”. Also decrypt it.
- Solution: Encryption
- Plain Tex: my name is jenney
- Split pair: my na me is je nx ne yx
- Here x is a filler character Key: monarchya

Now, generate key square:

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

- Now encrypt the plain text:

PT	MY	NA	ME	IS	JE	NX	NE	YX
CT	NC	AR	CL	SX	KF	AW	MG	BW

For MY: M and Y make a rectangle in the key square so take the opposite letter of M and Y as a cipher text
For NA: N and A are in same row, so take letter one step right of the letter N and A as a cipher text
For ME: M and E are in same column, so take letter down of the letter M and E as a cipher text Hence,
Plain Text: my name is jenney
Cipher Text: ncarclsxkfawmgbw

- For Decryption:
- Cipher Tex: ncarclsxkfawmgbw
- Split pair: nc ar cl sx kf aw mg bw
- Key: monarchya

Now, generate key square:

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

Now decrypt the cipher text:

CT	NC	AR	CL	SX	KF	AW	MG	BW
PT	MY	NA	ME	IS	JE	NX	NE	YX

For NC: N and C make a rectangle in the key square so take the opposite letter of N and C as a cipher text
For AR: A and R are in same row, so take letter one step left of the letter A and R as a cipher text
For CL: C and L are in same column, so take letter above of the letter C and L as a cipher text
Hence,
Cipher Text: ncarclsxkfawmgbw
Plain Text: mynameisjenxneyx
Now remove the filler character x
Plain Text: my name is jenney

Q. Find out cipher text of below plaintext using Playfair Cipher.
Plaintext: TREE IS GREEN, Keyword: ENVIRONMENT

Sol^{n}:

Keyword: ENVIRONMENT

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

Playfair matrix

Plaintext: TREE IS GREEN

Breaking the given plaintext as: TR EX EI SG RE EN

Ciphertext: BV VU NR QH EN NV

$\therefore Ciphertext = BVVUNRQHENNV$

Q. Decrypt the following ciphertext using Playfair cipher.

Keyword: KEYWORD Ciphertext: LCNKZKVFYOGQCEBW

Sol^{n}:

Keyword: KEYWORD

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

Ciphertext: LCNKZKVFYOGQCEBW

Breaking the given ciphertext as: LC NK ZK VF YO GQ CE BW

Plaintext: CO ME TO TH EW IN DO WX

$\therefore Plaintext = COME\ TO\ THE\ WINDOW$

MODERN CIPHERS

- Modern encryption is the key to advanced computer and communication security. Its foundation is based on various concepts of mathematics such as number theory, computational-complexity theory, and probability theory.
- Modern Ciphers can be divided in to two criteria:

- Based on Size
- Based on Key

- Based on Size:
- There are two types of Modern Cipher based on Size

- Stream Cipher
- Block Cipher

- **Stream Cipher:** In this scheme, the plain binary text is processed one bit or one byte at a time i.e. one bit of plain binary text is taken, and a series of operations is performed on it to generate one bit of ciphertext. Technically, stream ciphers are block ciphers with a block size of one bit.
- During Decryption also one bit or one byte of ciphertext is processed at a time. Example: Vigenere Cipher, Vernam Cipher etc.

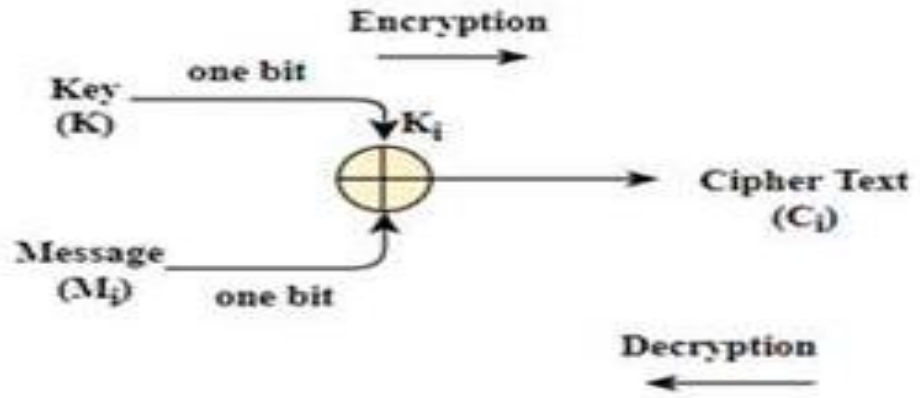


Fig: Stream Cipher

- Block Cipher: It is one that encrypt or decrypt a fixed size block of bits at one time
- In this scheme, the plain binary text is processed in blocks (groups) of bits at a time; i.e. a block of plain binary text bits is selected, a series of operations is performed on this block to generate a block of ciphertext bits. The number of bits in a block is fixed. For example, the schemes DES and AES have block sizes of 64 and 128, respectively. Example: DES, AES etc.

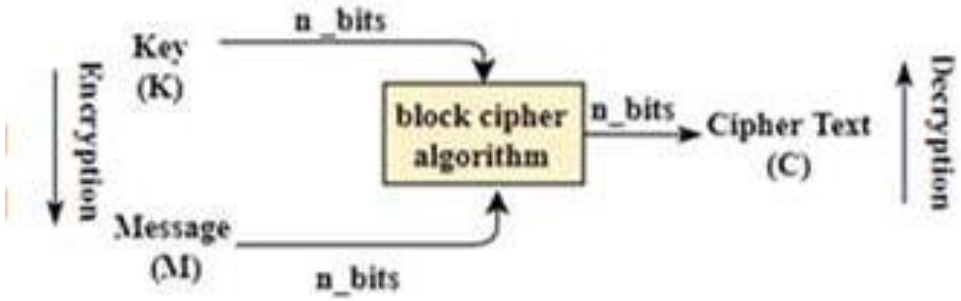


Fig: Block Cipher

Stream Cipher vs. Block Cipher

Stream Cipher	Block Cipher
1. Each time a bit is encrypted or decrypted at a time.	**1.** Each time a block of bits is encrypted or decrypted.
Advantages:	***Disadvantages:***
2. Faster than block cipher.	**2.** Slower than stream cipher.
3. Low error propagation i.e. an error in bit can only damage that bit.	**3.** High error propagation i.e. an error in bit can corrupt the whole block.
Disadvantages:	***Advantages:***
4. Low diffusion i.e. a secrecy of a bit is dependent only on a single bit.	**4.** High diffusion i.e. a secrecy of a single bit is depend on a whole block.
5. Susceptibility to malicious insertion and modification.	**5.** Immunity to insertion of symbol.
6. ***Example:*** One time pad	**6.** ***Example:*** DES (Data Encryption Standard)

**Confusion* *is the encryption technique where relationship between plain text and key make complex.*

**Diffusion* *is the encryption technique where cipher text make complex.*

- Based On Keys:
- Based on Keys there are two types of Modern Ciphers
- Symmetric Cipher
- Asymmetric Cipher

Fig: Symmetric Ciphers

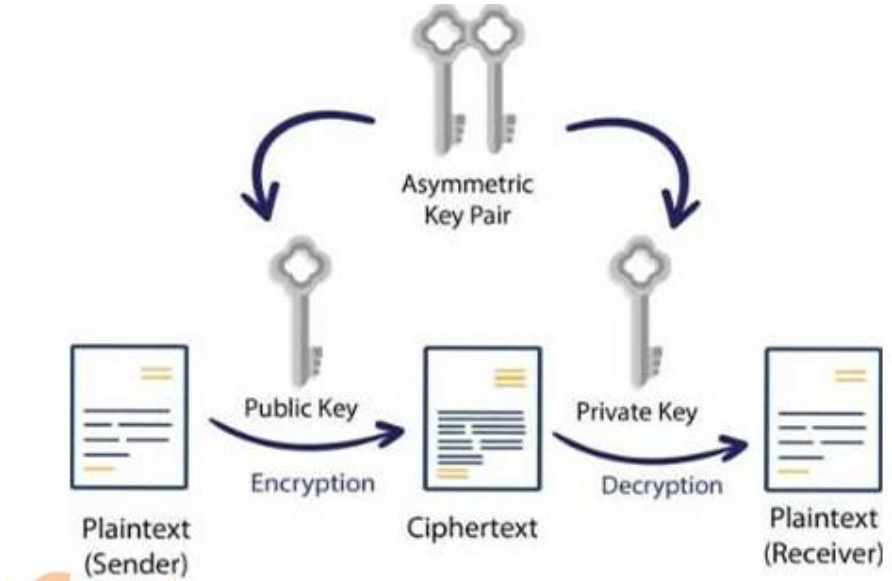


Fig: Asymmetric Cipher

- **Symmetric Cipher:**
- Symmetric encryption also called private key cryptography, involves the use of one key for both encryption and decryption. The plaintext is encrypted using a secret key to generate a cipher text. To decrypt the cipher text in to original plain text need that same secret key which is used for encryption.
- This works well for data that is being stored and needs to be decrypted at a later date. The use of just one key for both encryption and decryption reveals an issue, as the compromise of the key would lead to a compromise of any data the key has encrypted. This also does not work for data-in-motion. To overcome this issue asymmetric encryption comes in.
- **Asymmetric Cipher:**
- Asymmetric encryption also called public key cryptography, works with a pair of keys. The beginning of asymmetric encryption involves the creation of a pair of keys, one of which is a public key, and the other which is a private key.
- The public key is accessible by anyone, while the private key must be kept a secret from everyone by the creator of the key. This is because encryption occurs with the public key, while decryption occurs with the private key. The recipient of the sensitive data will provide their public key to the sender, which will be used to encrypt the data. This ensures that only the recipient can decrypt the data, with their own private key.

Symmetric vs Asymmetric Cryptography

Symmetric Cryptography	***Asymmetric Cryptography***
In symmetric cryptography, single or same key is used for both encryption as well as decryption.	In asymmetric cryptography, two separate keys are used, one for encryption and the other for decryption.
It is faster than asymmetric cryptography.	Because of different keys used for encryption and decryption, it is slower than symmetric cryptography.
It utilizes less resources as compared to asymmetric cryptography.	It uses more resources as compared to symmetric cryptography.
Its strength of secrecy is weak.	Its strength of secrecy is relatively strong.
Key must be kept secret.	Public key is publically available.
E.g. Caesar, DES	E.g. RSA

SYMMETRIC ENCRYPTION

- Symmetric Encryption use same secret key for encryption and decryption. There are various ways of implementation of symmetric encryption techniques. Some of them are described below:

FEISTEL CIPHER STRUCTURE:

- The Feistel cipher is a design model or structure used to build various symmetric block ciphers, such as DES. This design model can have invertible, non-invertible, and self-invertible components. Additionally, the Feistel block cipher uses the same encryption and decryption algorithms.
- The Feistel structure is based on the Shannon structure proposed in 1945, demonstrating the confusion and diffusion implementation processes**. Confusion** produces a complex relationship between the ciphertext and encryption key, which is done by using a substitution algorithm. On the other hand, **diffusion** creates a complex relationship between plain text and cipher text by using a permutation algorithm.
- The Feistel cipher proposed the structure that implements substitution and permutation alternately. Substitution replaces plain text elements with ciphertext. Permutation changes the order of the plain text elements rather than being replaced by another element as done with substitution.

- Fiestel cipher is a type of block cipher design, not a specific cipher.
- In a fiestel cipher, the plaintext block P is divided into left and right halves: $P = (L_0, R_0)$. Then the two halves pass through n rounds of processing then combine to produce the cipher block.

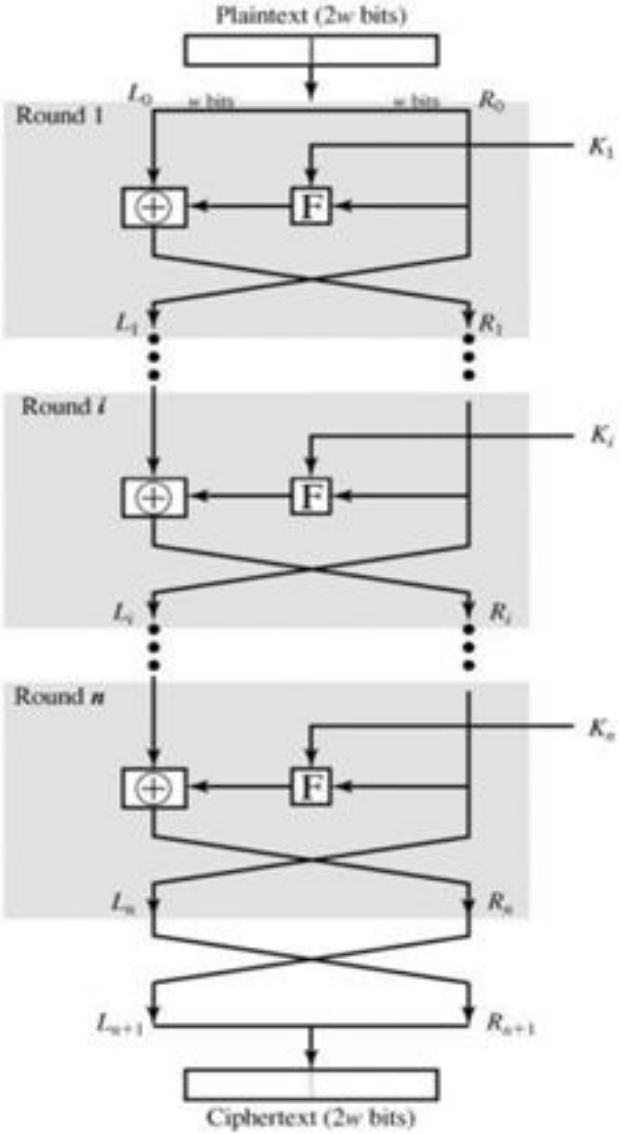
  For each round $i = 1, 2, \ldots\ldots n$ new left and right halves are computed according to the rules:

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

  Where, F is round function and K_i is the subkey for round i.

  The subkey is derived from the original key K_i according to a specified key schedule algorithm. Finally the ciphertext C is the output of the final round.

  $$C = (L_n, R_n)$$
- All rounds have the same structure.
- A **substitution** is performed on the left half of the data. This is done by applying a round function F to the right half of data followed by the XOR of the output of that function and left half of data.
- The **permutation** steps at each round swaps the modified L and unmodified R.
- The combination of substitution and permutation is called a round.



Decryption

- The process of decryption is similar to encryption with reversed keys.

  Start with ciphertext $C = (L_n, R_n)$

  For each round $i = n, n-1, \ldots\ldots\ldots, 1$ compute:

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

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

  Where, F is round function and K_i is subkey.

  Plaintext: $P = (L_0, R_0)$

Design Features/Principles of Fiestel Network:

- ***Block size:*** increasing size improves security, but reduced encryption/decryption speed.
- ***Key size:*** increasing key size improves security, makes exhaustive key searching harder, but may slow cipher.
- ***Number of rounds:*** increasing number of rounds increases security, but slows cipher.
- ***Sub-key generation algorithm:*** greater complexity in this algorithm can make analysis harder, but slows cipher.
- ***Round function:*** greater complexity can make analysis harder, but slows cipher.
- ***Fast software en/decryption & ease of analysis:*** are more recent concerns for practical use and testing.

First, consider the encryption process. We see that

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \times F(RE_{15}, K_{16})$$

On the decryption side,

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \times F(RD_0, K_{16})$$

$$= RE_{16} \times F(RE_{15}, K_{16})$$

$$= [LE_{15} \times F(RE_{15}, K_{16})] \times F(RE_{15}, K_{16})$$

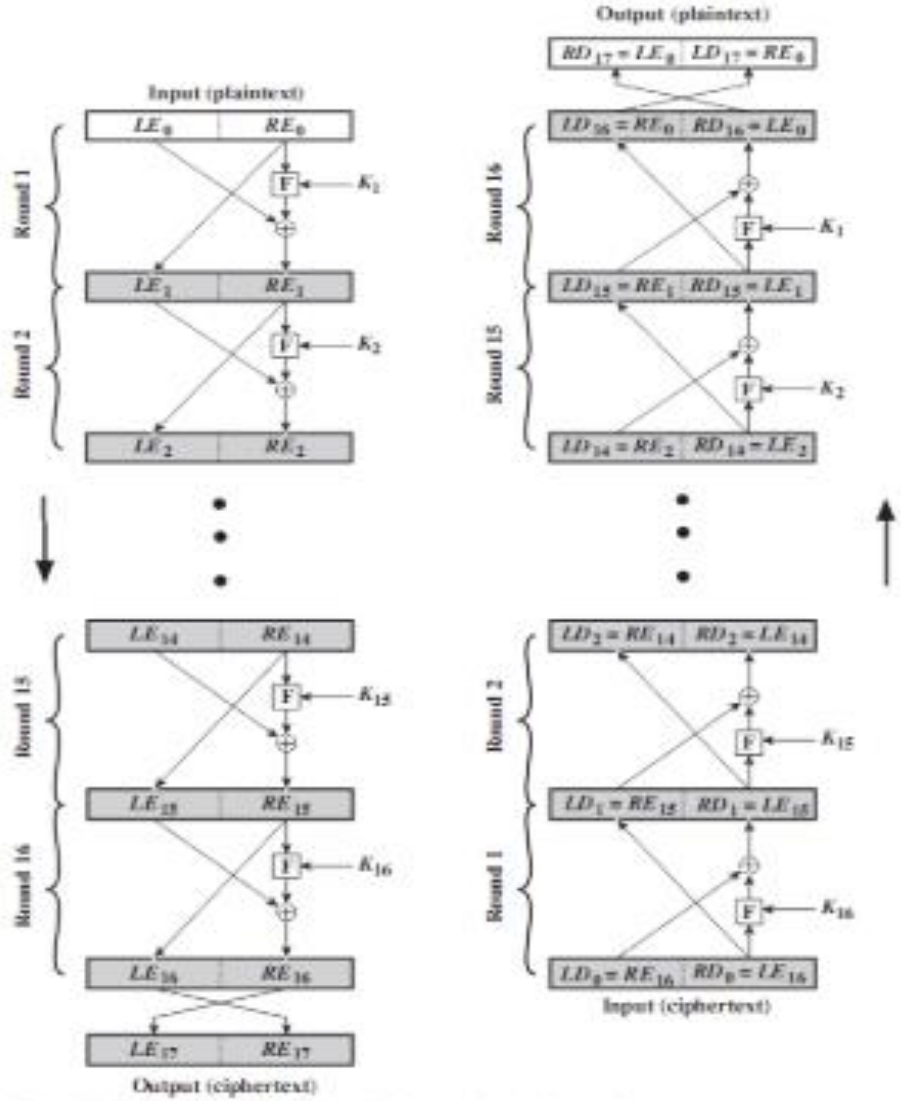
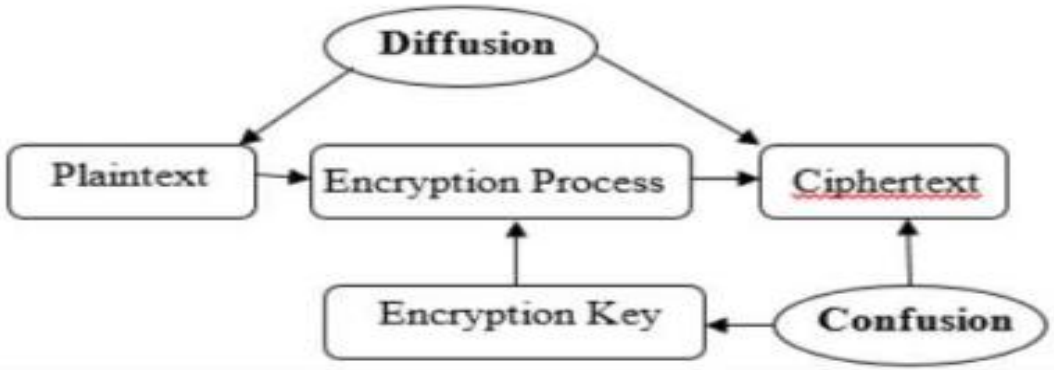


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

CONFUSION AND DIFFUSION

- **Confusion** refers to making the relationship between the key and ciphertext as complex and involved as possible.
  - Each bit of the ciphertext should depend on several part of the key, obscuring the connections between the two.
  - E.g. S-box or substitution cipher.
- **Diffusion** means any of the characters in the plaintext is changed, then simultaneously several characters of the ciphertext should also be changed. Similarly, if the characters of ciphertext is changed then simultaneously several characters of plaintext should be changed.
  - Diffusion hides the relation between the ciphertext and the plaintext.
  - E.g. P-box or transposition cipher.



DATA ENCRYPTION STANDARDS(DES)

- DES is symmetric-key algorithm which is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences.
- The key length is 56 bits. There are certain machines that can be used to crack the DES algorithm, therefore, the popularity of DES has been found slightly on the decline.

- The overall scheme for DES encryption is illustrated in figure.
- As with any encryption scheme, there are two inputs to the encryption function:
- The plaintext to be encrypted and The key.
- In DES, the plaintext must be 64 bits in length and the key is 56 bits in length.
- Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.

1. Initial permutation (Permute 64 bits in a block)
2. Sixteen rounds of the same function, which involves both permutation and substitution function (Apply given operation 16 times on the 64 bits)
3. Inverse permutation (Permute the 64 bits using the inverse of initial permutation)

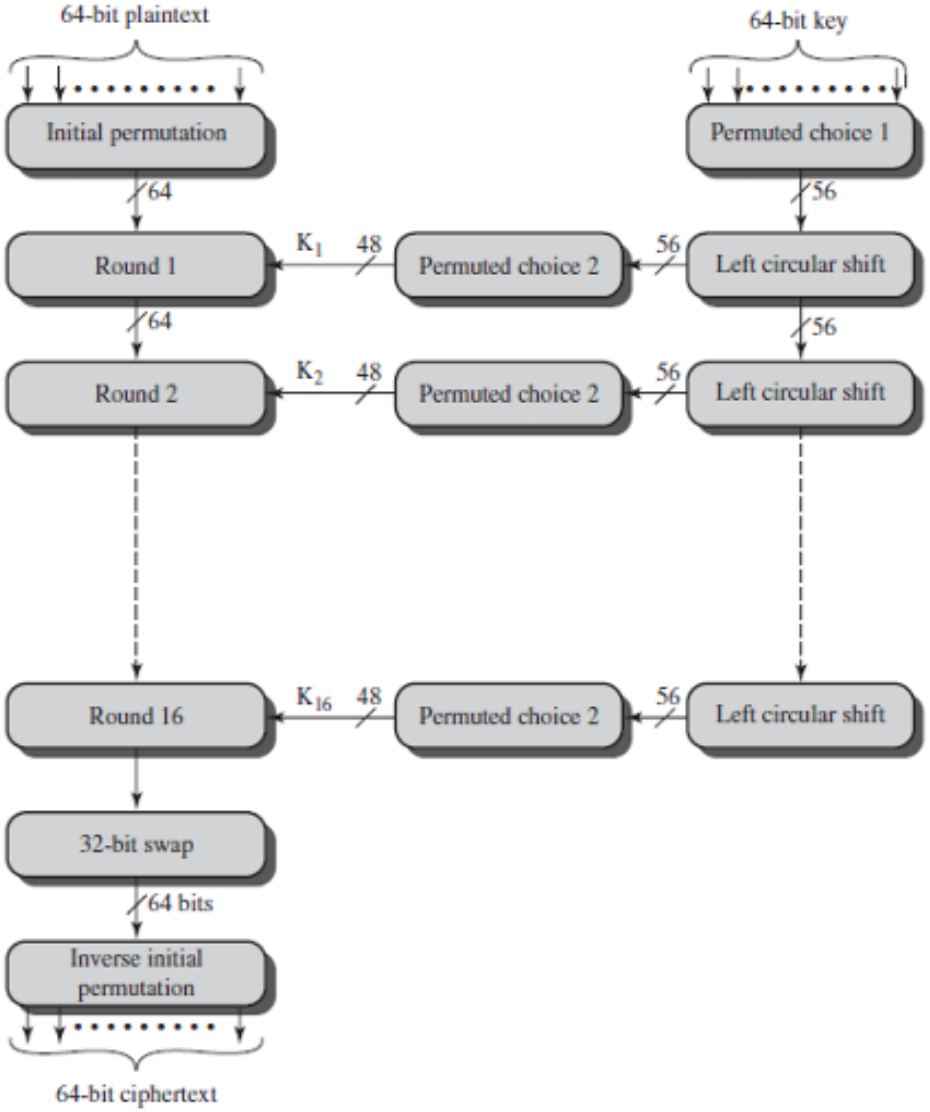


Figure: General Depiction of DES Encryption Algorithm

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.

→ This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions.

- The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key.

- The left and right halves of the output are swapped to produce the preoutput.

→ Finally, the preoutput is passed through a permutation [IP -1] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.

- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.
- Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a subkey (Ki) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

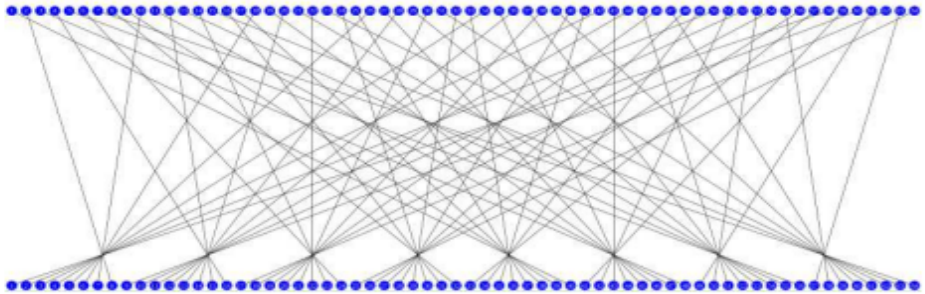
Key Generation

In DES encryption, the round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.

Initially, 56 bits of the key are selected from the initial 64-bit by Permuted Choice 1 (PC-1), the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves (C_0 &D_0); each half is thereafter treated separately. In successive rounds, both halves(C_{i-1}&D_{i-1}) are rotated left by one or two bits specified for each round, and then 48 subkey bits are selected by Permuted Choice 2, PC-2 (24 bits from the left half, and 24 from the right) that serves as input to the function $F(R_{i-1}, K_i)$.

Initial Permutation (IP)

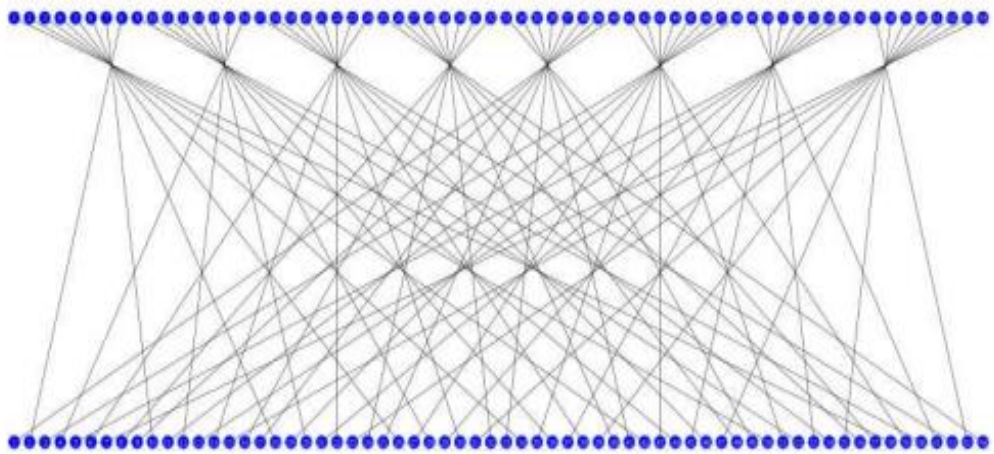
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

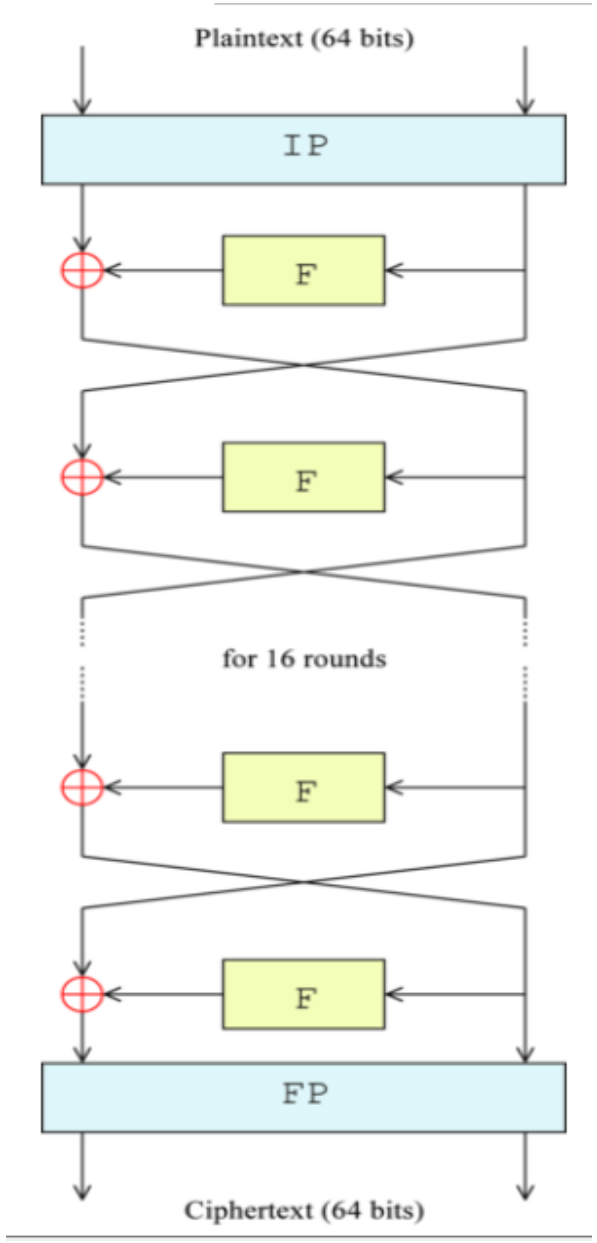


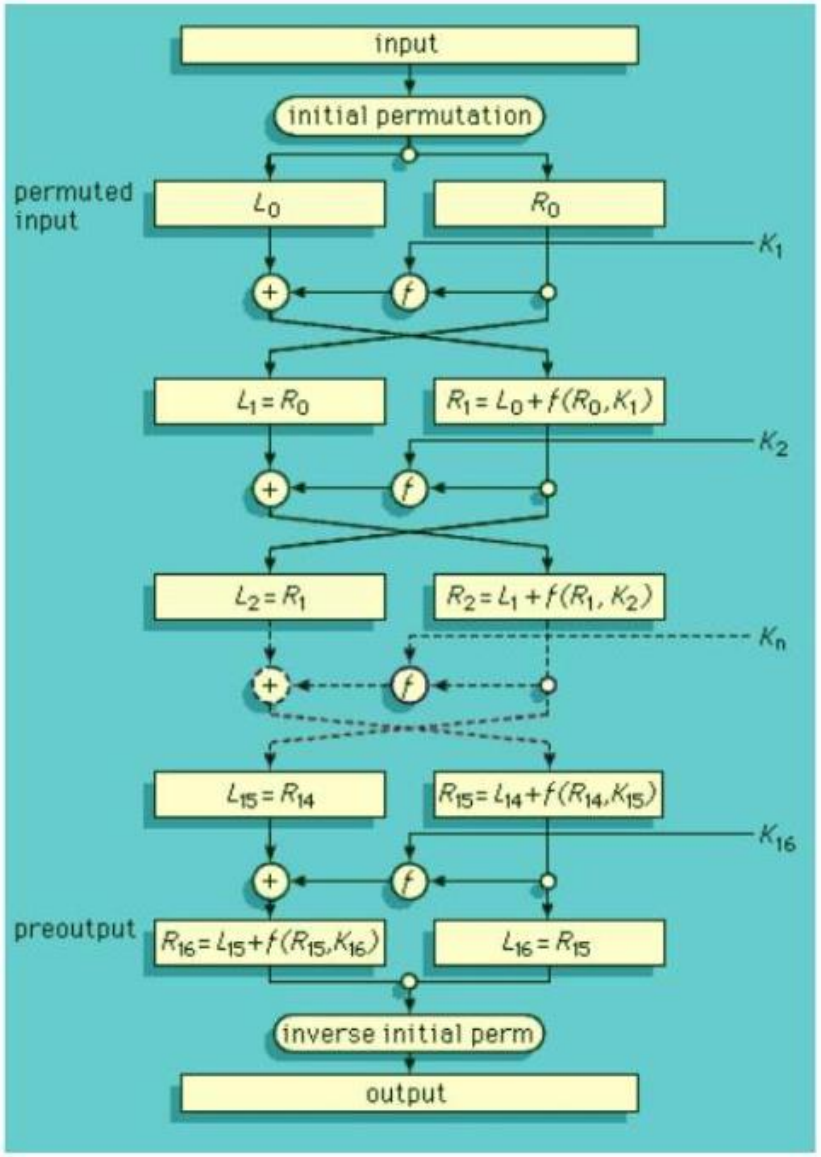
- This table specifies the input permutation on a 64-bit block.
- The meaning is as follows:
  - the first bit of the output is taken from the 58th bit of the input; the second bit from the 50th bit, and so on, with the last bit of the output taken from the 7th bit of the input.

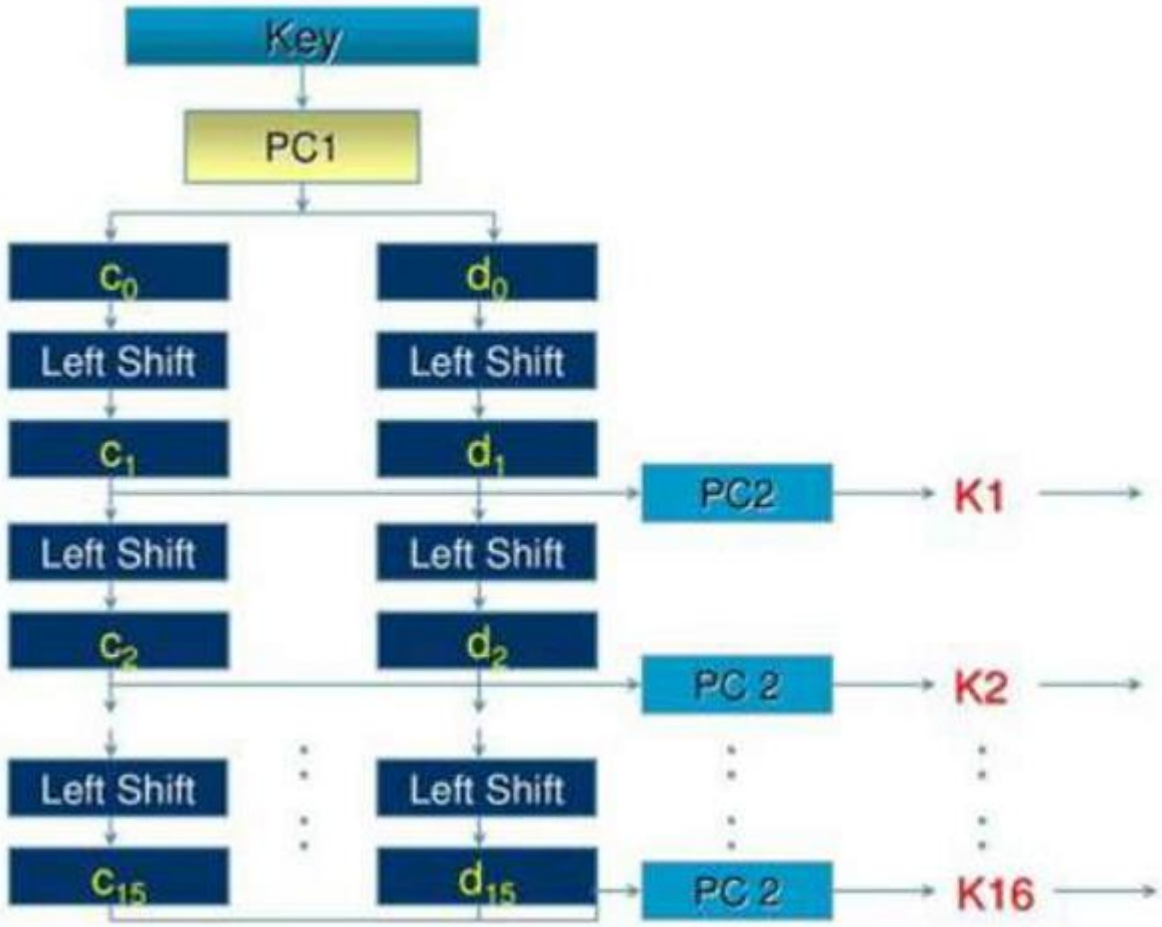
Final Permutation (IP^{-1})

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




Plaintext (64 bits)
IP
F
F
for 16 rounds
F
F
FP
Ciphertext (64 bits)


input
initial permutation
permuted input
L_0
R_0
K_1
$L_1 = R_0$
$R_1 = L_0 + f(R_0, K_1)$
K_2
$L_2 = R_1$
$R_2 = L_1 + f(R_1, K_2)$
K_n
$L_{15} = R_{14}$
$R_{15} = L_{14} + f(R_{14}, K_{15})$
K_{16}
preoutput
$R_{16} = L_{15} + f(R_{15}, K_{16})$
$L_{16} = R_{15}$
inverse initial perm
output



Shifting

Rounds	***Shift***
1, 2, 9, 16	One bit
Others	Two bit

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Permuted Choice 1 and 2 (PC-1, PC-2)

Parity-check bits (namely, bits 8,16, 4,32,40,48,56,64) are not chosen, they do not appear in **PC-1**

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

PC-2 selects the 48-bit subkey for each round from the 56-bit key-schedule state

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

In DES encryption, the round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.

Initially, 56 bits of the key are selected from the initial 64-bit by Permuted Choice 1 (PC-1), the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves (C_0 &D_0); each half is thereafter treated separately. In successive rounds, both halves(C_{i-1}&D_{i-1}) are rotated left by one or two bits specified for each round, and then 48 subkey bits are selected by Permuted Choice 2, PC-2 (24 bits from the left half, and 24 from the right) that serves as input to the function $F(R_{i-1}, K_i)$.

Example: Let **K** be the hexadecimal key **K** = 133457799BBCDFF1. This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

The DES algorithm uses the following steps:

Step 1: Create 16 sub-keys, each of which is 48-bits long.

The 64-bit key is permuted according to the following table, **PC-1**. Since the first entry in the table is "57", this means that the 57th bit of the original key **K** becomes the first bit of the permuted key **K**+. The 49th bit of the original key becomes the second bit of the permuted key. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

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

Example: From the original 64-bit key

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, C_0 and D_0, where each half has 28 bits.

Example: From the permuted key **K+**, we get

C_0 = 1111000 0110011 0010101 0101111

D_0 = 0101010 1011001 1001111 0001111

Example: From original pair pair C_0 and D_0 we obtain:

C_0 =1111000011001100101010101111

D_0 = 0101010101100110011110001111

C_1 =1110000110011001010101011111

D_1 = 1010101011001100111100011110

C_2 =1100001100110010101010111111

D_2 = 0101010110011001111000111101

C_3 =0000110011001010101011111111

D_3 = 0101011001100111100011110101

C_4 =0011001100101010101111111100

D_4 = 0101100110011110001111010101

PC-2

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

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

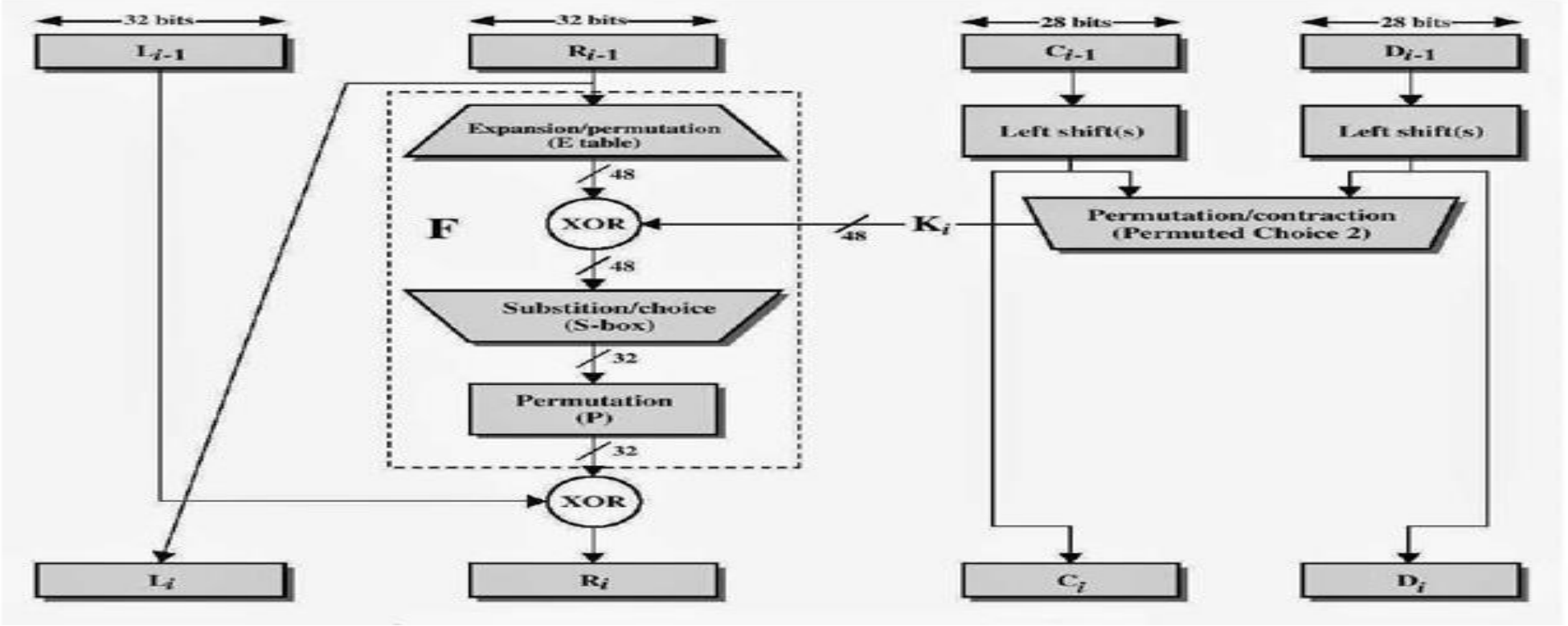
Example: For the first key we have C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

which, after we apply the permutation **PC-2**, becomes

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

Single Round of DES Algorithm

The following figure shows a closer view of algorithms for a single iteration. The 64bit permuted input passes through 16 iterations, producing an intermediate 64-bit value at the conclusion of each iteration.



The left hand output of an iteration (L_i) is equal to the right hand input to that iteration R_{i-1}. The right hand output R_i is exclusive OR of L_{i-1} and a complex function F of K_i and R_{i-1}. The function F can be depicted by the following figure. $S1, S2, \ldots\ \ldots\ \ldots, S8$ represent the "***S-boxes***" , which maps each combination of 48 input bits into a particular 32 bit pattern.

Internal Structure of DES

The building blocks of DES :

- The initial and final permutation
- The F–function
- The key schedule

Initial and Final Permutation

- Initial permutation IP and the final permutation IP−1 are bitwise permutations.
- A bitwise permutation can be viewed as simple crosswiring
- The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other.
- They have no cryptography significance in DES

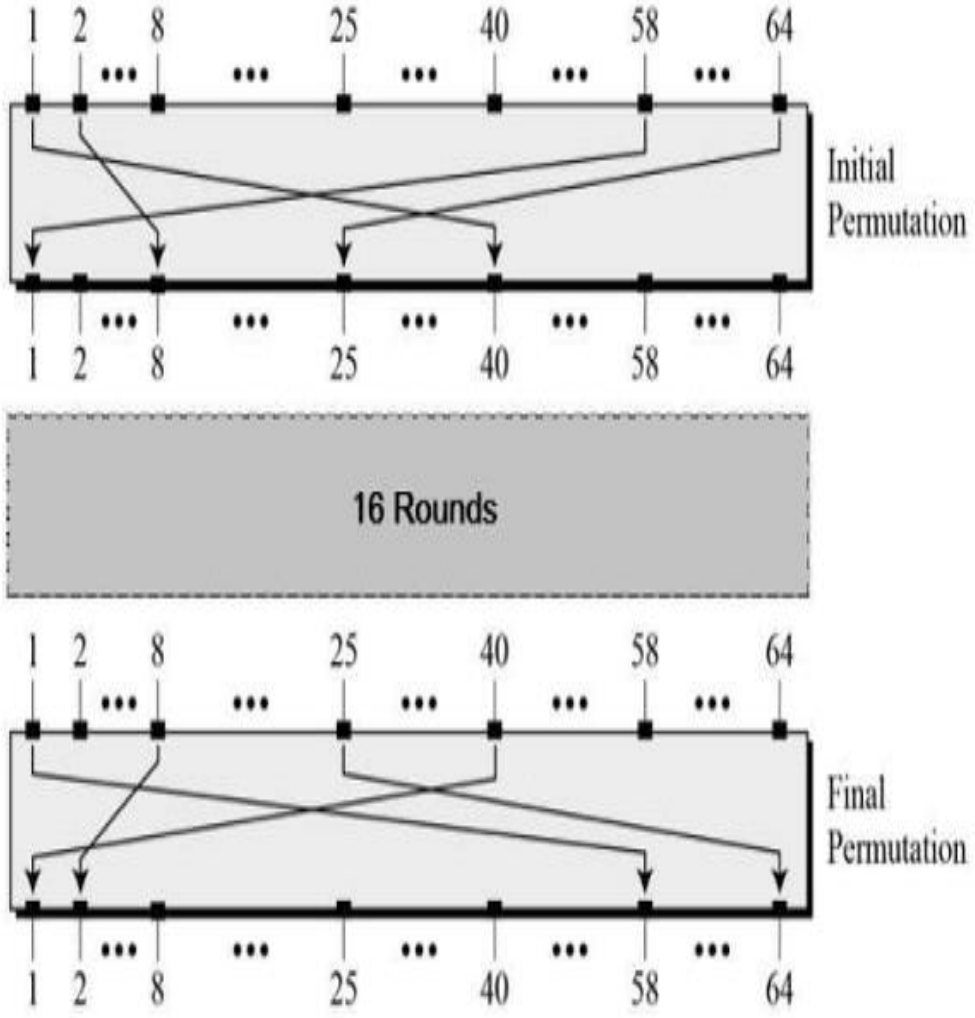


Figure: Examples for the bit swaps of the initial and final permutation

→ The permutation rules for these P-boxes are shown below. Each box can be thought of as a 64-element array

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

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

This table should be read from left to right, top to bottom. The table (IP) indicates that input bit 58 is mapped to output position 1, input bit 50 is mapped to the second output position, and so forth.

The F-Function

- The F -function plays a crucial role for the security of DES.
- In round i it takes the right half R_{i-1} of the output of the previous round and the current round key k_i as input.
- The output of the F -function is used as an XOR-mask for encrypting the left half input bits L_{i-1}

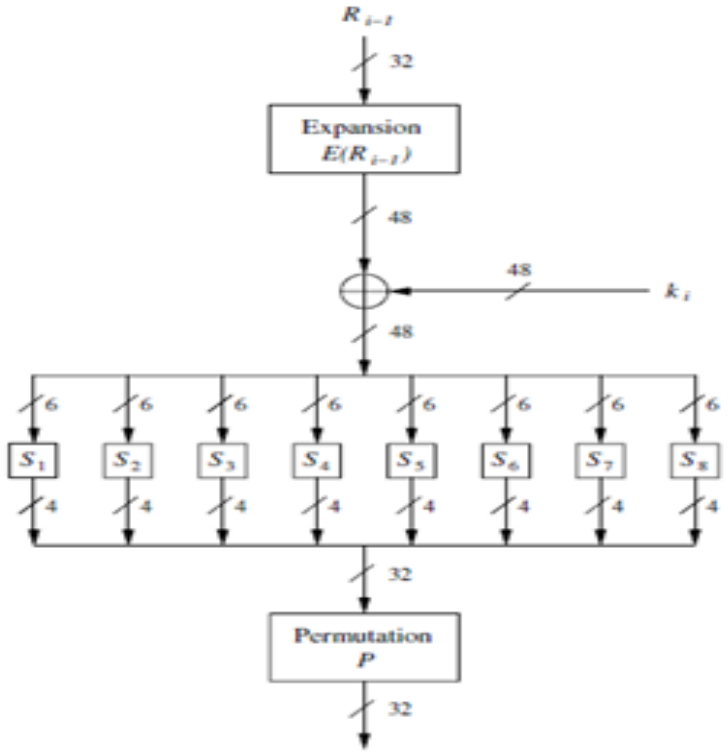


Figure : Block diagram of the F –function in DES

Four steps of F function

- Expansion E
- XOR with round key
- S-Box substitution
- Permutation

→ First, the 32-bit input is expanded to 48 bits by partitioning the input into eight 4-bit blocks and by expanding each block to 6 bits. This happens in the E-box (Expansion Box).

→ E-box is a special type of permutation. The first block consists of the bits (1,2,3,4), the second one of (5,6,7,8), etc. The expansion to six bits can be seen following figure

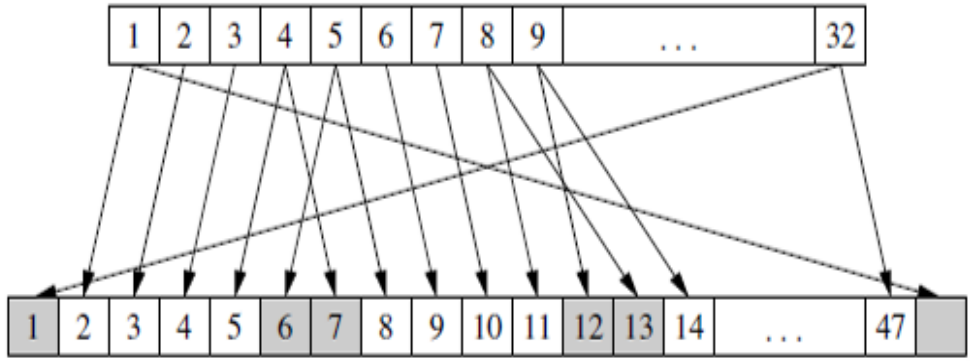


Figure: Examples for the bit swaps of the expansion function E

→ *Main purpose of E- Box: increases diffusion*

→ As can be seen from the following table , exactly 16 of the 32 input bits appear twice in the output

Table: Expansion permutation E

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

→ Next, the 48-bit result of the expansion is XORed with the round key k_i, and the eight 6-bit blocks are fed into eight different **substitution boxes**, which are often referred to as **S-boxes.**

→ Each ***S-box is a lookup table that maps a 6-bit input to a 4-bit output.***

→ *S-Box is the "heart" of DES!*

→ *Main Purpose of S-Box: provides confusion*

→ Each S-box contains 2^6 =64 entries, which are typically represented by a table with 16 columns and 4 rows. Each entry is a 4-bit value.
(Note: All S-boxes are listed in tables below)

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

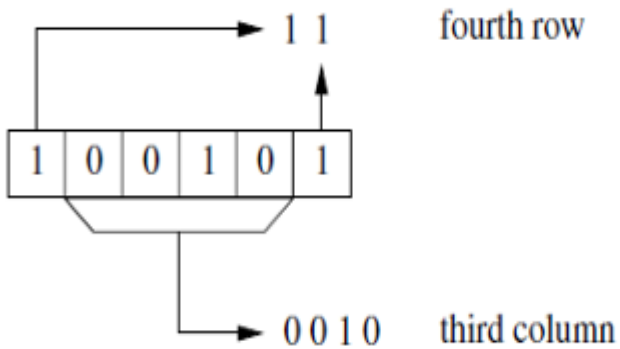
→ The tables are to be read as follows :

- The most significant bit (MSB) and the least significant bit (LSB) of ***each 6-bit input select the row of the table***, while the ***four inner bits select the column***.
- The integers 0,1,. . . ,15 of each entry in the table represent the decimal notation of a 4-bit value.

S-box S_1

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Example of the decoding of the input 100101_2 by S-box 1



⇒ The S-box input b = $(100101)_2$ indicates the row $11_2 = 3$ (i.e., fourth row, numbering starts with 00_2) and the column $0010_2 = 2$ (i.e., the third column).

⇒ If the input b is fed into S-box 1, the output is

$S_1(37 = 100101_2) = 8 = 1000$

Example (S1)

Row #	S_1	1	2	3	...	7	...	15	Column #
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S(i, j) <16, can be represented with 4 bits

Example: B =101111

b_1b_6 = 11 = row 3

$b_2b_3b_4b_5$= 0111 = column 7

C=7=0111

Another example: B=011011, C=?

→ Finally, the 32-bit output is permuted bitwise according to the P permutation, which is given in following table

Table: The permutation P within the f -function

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

.

- ✓ Unlike the initial permutation IP and its inverse IP−1, the permutation P introduces *diffusion* because ***the four output bits of each S-box are permuted in such a way that they affect several different S-boxes in the following round***.
- ✓ The diffusion caused by the expansion, S-boxes and the permutation P guarantees that every bit at the end of the fifth round is a function of every plaintext bit and every key bit. This behavior is known as the **avalanche effect**

The Key Schedule

- The key schedule derives *16 round keys* k_i, each consisting of 48 bits, from the original 56-bit key. Another term for round key is *subkey*

  (Note: The DES input key is often stated as 64-bit, where every eighth bit is used as an odd parity bit over the preceding seven bits. It is not quite clear why DES was specified that way. In any case, the eight parity bits are not actual key bits and do not increase the security. DES is a 56-bit cipher, not a 64-bit one)

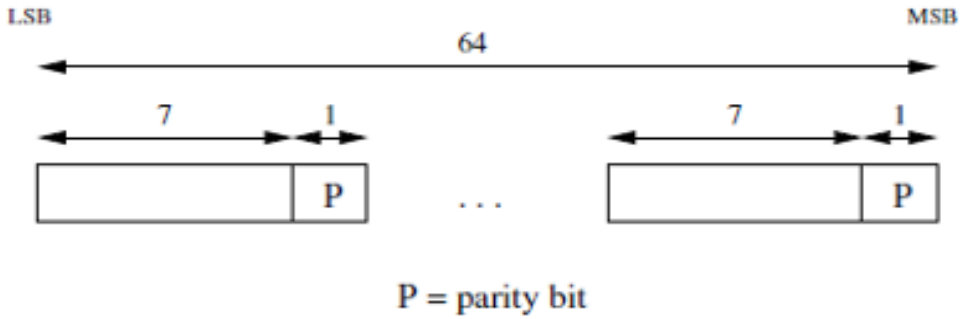


Figure: Location of the eight parity bits for a 64-bit input key

- The 64-bit key is first reduced to 56 bits by ignoring every eighth bit, i.e., the parity bits are stripped in the initial PC−1 permutation. Again the parity bits certainly do not increase the key space! The name PC−1 stands for "permuted choice one". The exact bit connections that are realized by PC−1 are given in following table.

Table : Initial key permutation PC−1

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

- The resulting 56-bit key is split into two halves C_0 and D_0, and the actual key schedule starts as shown in following figure.

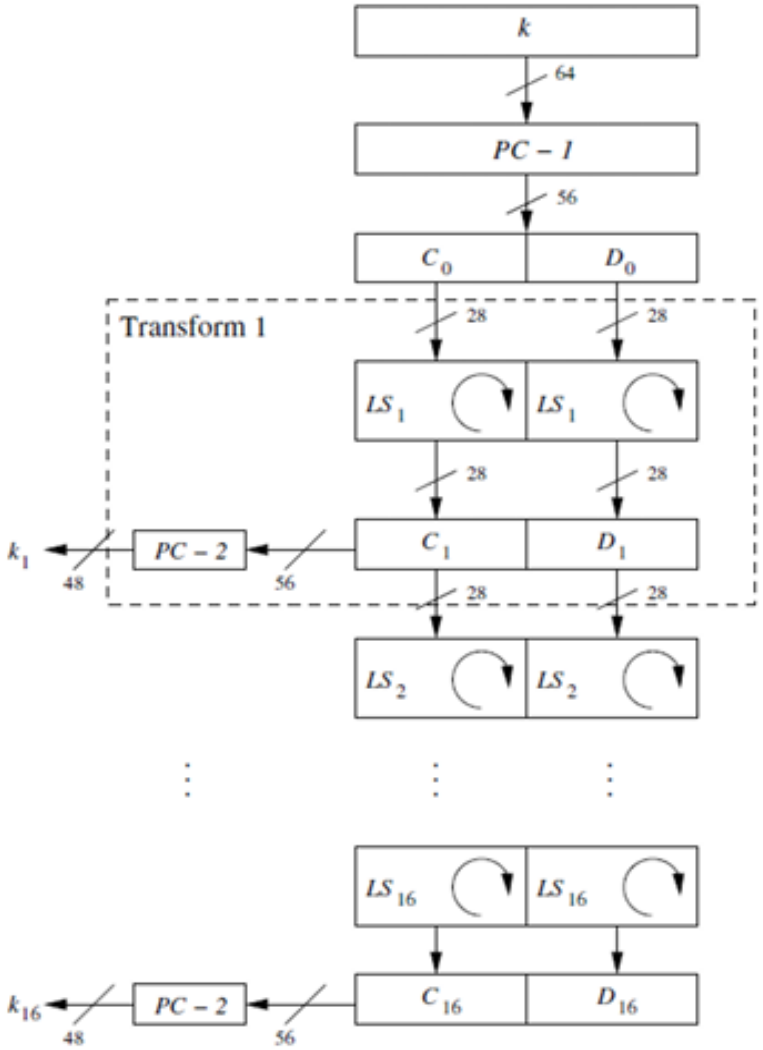


Figure: Key schedule for DES encryption

- The two 28-bit halves are cyclically shifted, i.e., rotated, left by one or two bit positions depending on the round i according to the following rules:
  - In rounds i = 1,2,9, 16, the two halves are rotated left by one bit.
  - In the other rounds where i ≠ 1,2,9, 16, the two halves are rotated left by two bits.
- The rotations only take place within either the left or the right half.
- The total number of rotation positions is 4 × 1+12 × 2 = 28. This leads to the interesting property that $C_0 = C_{16}$ and $D_0 = D_{16}$
- To derive the 48-bit round keys k_i, the two halves are permuted bitwise again with PC−2, which stands for "permuted choice 2". PC−2 permutes the 56 input bits coming from C_i and D_i and ignores 8 of them. The exact bit-connections of PC−2 are given in following table.

Table : Round key permutation PC−2

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

- Every round key is a selection of 48 permuted bits of the input key k. The key schedule is merely a method of realizing the 16 permutations systematically.

- **DES Decryption Cryptography**
- As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.
- Compared to encryption, only the key schedule is reversed, i.e., in decryption round 1, subkey 16 is needed; in round 2, subkey 15; etc.
- Thus, when in decryption mode, the key schedule algorithm has to generate the round keys as the sequence k16,k15, . . . ,k1.
- Additionally, the initial and final permutations are reversed.

- DES Algorithm Steps:
- Step 1: The process begins with the 64-bit plain text block getting handed over to an initial permutation (IP) function.
- Step 2: The initial permutation (IP) is then performed on the plain text.
- Step 3: Next, the initial permutation (IP) creates two halves of the permuted block, referred to as Left Plain Text (LPT) and Right Plain Text (RPT).
- Step 4: Each LPT and RPT goes through 16 rounds of the encryption process. Here five more sub-stages are performs during encryption process

1. Key transformation
2. Expansion permutation
3. S-Box permutation
4. P-Box permutation
5. XOR and swap

- Step 5: Finally, the LPT and RPT are rejoined, and a Final Permutation (FP) is performed on the newly combined block. The result of this process produces the desired 64-bit ciphertext.
- For decryption, we use the same algorithm, and we reverse the order of the 16 round keys.

- Advantages of the DES algorithm:
- It is set as a standard by the US government.
- When compared to the software, it works faster on hardware.
- Triple DES, used a 168-bit key which is very hard to crack.
- Disadvantages of the DES algorithm:
- Weakly secured algorithm.
- There is a threat from Brute force attacks.
- A DES cracker machine known as Deep Crack is available in the market.

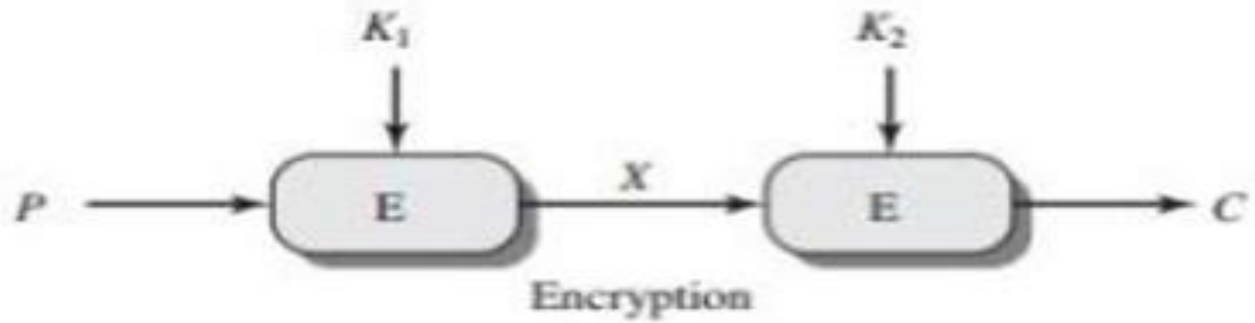
Double DES

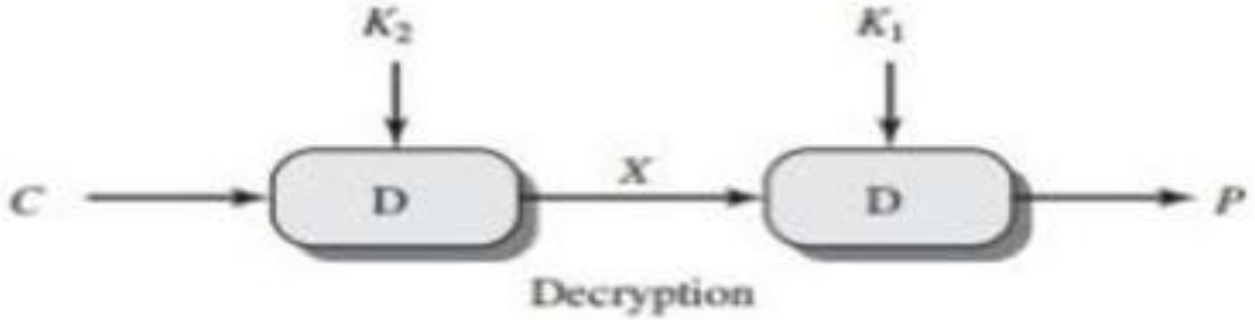
- It does twice what DES normally does only once.
- It uses two keys K_1 & K_2 and encrypt the text using the two keys.

$$C = E_{K_2}(E_{K_1}(P))$$

- To decrypt simply use DES decryption twice.

$$C = D_{K_1}(D_{K_2}(C))$$



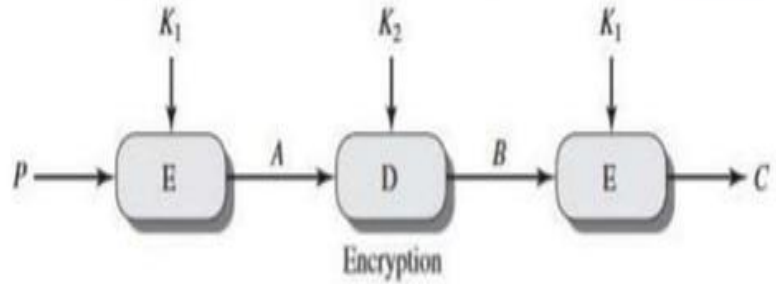


Triple DES

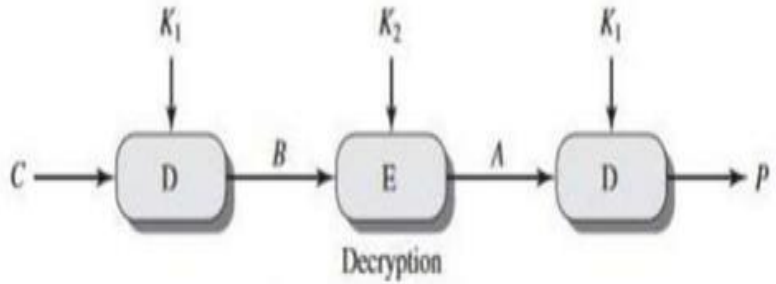
- ***With Two Keys:*** It uses three stages of DES for encryption and decryption. The 1st & 3rd stages use K_1 key and 2nd stage uses K_2 key. To make triple DES compatible with single DES, the middle stage uses decryption in the encryption side and encryption in the decryption side.

  Encryption: $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$

  Decryption: $P = D_{K_1}(E_{K_2}(D_{K_1}(C)))$



Encryption

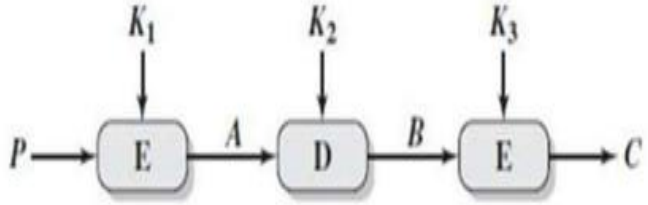


Decryption

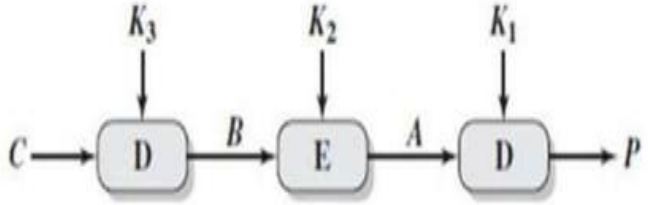
- ***With Three Keys:*** It uses three stages of DES for encryption and decryption with three different keys.

  Encryption: $C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$

  Decryption: $P = D_{K_1}(E_{K_2}(D_{K_3}(C)))$



Encryption



Decryption

BASIC CONCEPT OF FIELDS: GROUPS, RINGS, FIELDS

Groups

- denoted by $\{G, \cdot\}$, where $\cdot$ is generic symbol and can be a binary symbol.
- is a set of elements with a binary operation, such that following axioms are obeyed:
  - $A_1: closure \rightarrow if\ a, b \in G, then\ a \cdot b \in G$
  - $A_2: Associative \rightarrow a \cdot (b \cdot c) = (a \cdot b) \cdot c\ for\ all\ a, b, c \in G$
  - $A_3: Identity \rightarrow a \cdot e = e \cdot a\ for\ all\ a \in G$
  - $A_4: Inverse \rightarrow a \cdot a' = a' \cdot a = e\ for\ each\ a \in G$
  - $A_5: Commutative \rightarrow a \cdot b = b \cdot a\ for\ all\ a, b \in G$

$\rightarrow$ If A_1, A_2, A_3, A_4 satisfies = Group

$\rightarrow$ If A_1, A_2, A_3, A_4, A_5 satisfies = Abelian Group

Note: *Binary operation (.) includes (+, -) operation. Actually (-) operation is perform by (+) operation. For example, a - b = a + (-b)*

Rings

- denoted by $\{R, +, *\}$
- is a set of elements with two binary operations addition and multiplication, such that following axioms are obeyed:
  - $Abelian\ group\ under\ addition\ [A_1 - A_5]$
  - $M_1: Closure\ under\ multiplication \rightarrow if\ a, b \in R, then\ ab \in R$
  - $M_2: Associativity\ of\ multiplication \rightarrow a(bc) = (ab)c\ for\ all\ a, b, c \in R$
  - $M_3: Distributive \rightarrow a(b + c) = ab + ac\ for\ all\ a, b, c \in R$
  - $M_4: Commutative \rightarrow ab = ba\ for\ all\ a, b \in R$
  - $M_5: Multiplicative\ identity \rightarrow ae = ea = a\ for\ all\ a \in R$
  - $M_6: No\ zero\ divisor \rightarrow if\ ab = 0\ then\ either\ a = 0\ or\ b = 0, a, b \in R$

$\rightarrow$ If $A_1 - M_4$ satisfies = Commutative ring

$\rightarrow$ If $A_1 - M_6$ satisfies = Integral domain

Field:

A field F is indicated by **{F, +, *},** It is a set of elements with two binary operations known as addition and multiplication where subtraction is done using addition and division is done using multiplication, including for all a, b, c in F the following axioms are kept:

- F1 is an integer domain that is F satisfies axioms A1 through A5 and M1 through M6.
- **(M7): Multiplication inverse**: For each a in F, except 0, there is an element a^{-1} in F such that $aa^{-1} = (a^{-1})a=1$.

Example of Field is:

- Set of all **Rational numbers** (Q)
- Set of all Integers (Z) is not a Field. Because suppose 5 is an integer and multiplicative inverse of 5 is 1/5 and 1/5 is not in a set of integers.

MODULAR ARITHMETIC

- If **‘*a*’** is an integer and **‘*n*’** is a positive integer then we define ***a* mod *n*** to be the remainder when **‘*a*’** is divided by **‘*n*’**. The integer **‘*n*’** is called the **modulus**.
- It can be denoted as (a | n) or (a mod n)
- **Example:**
- **10 mod 3 = 1,** here 10 is divided by 3 then, quotient is 3 remainder is 1
- **5 mod 7 = 5,** here 5 is divided by 7 then, quotient is 0 and remainder is 5
- **-5 mod 8 = 3,** here integer is –ve so first add integer and modulus (-5+8 = 3) then perform (3 mod 8) = 3
- **-11 mod 7 = 3,** here integer is –ve so first add integer and modulus (-11+7 = -4) then perform (-4 mod 7) again integer is –ve so add integer and modulus (-4+7 = 3) then perform (3 mod 7) = 3

- ***Properties of Modular Arithmetic:***

- $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
- $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
- $[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$

- ***Congruence:***
- In cryptography, congruence (≡) instead of equality (=).
- **Example:**
- **15 ≡ 3 (mod 12),** this means 15 mod 12 = 3
- **23 ≡ 11 (mod 12),** this means 23 mod 12 = 11
- **10 ≡ -2 (mod 12),** this means 10 mod 12 = -2
- Or, this can also say as:
- **10 ≡ 10 (mod 12),** this means 10 mod 12 = 10

Two integers a and b are said to be congruent modulo n, if $\boldsymbol{a \equiv b (mod\ n)}$ i.e. when a is divided by n, we get remainder b.

E.g. $7 \equiv 2 (mod\ 5)$

$$a \equiv b (mod\ n) \Leftrightarrow (a - b)\ mod\ n = 0 \Leftrightarrow (a\ mod\ n) = (b\ mod\ n)$$

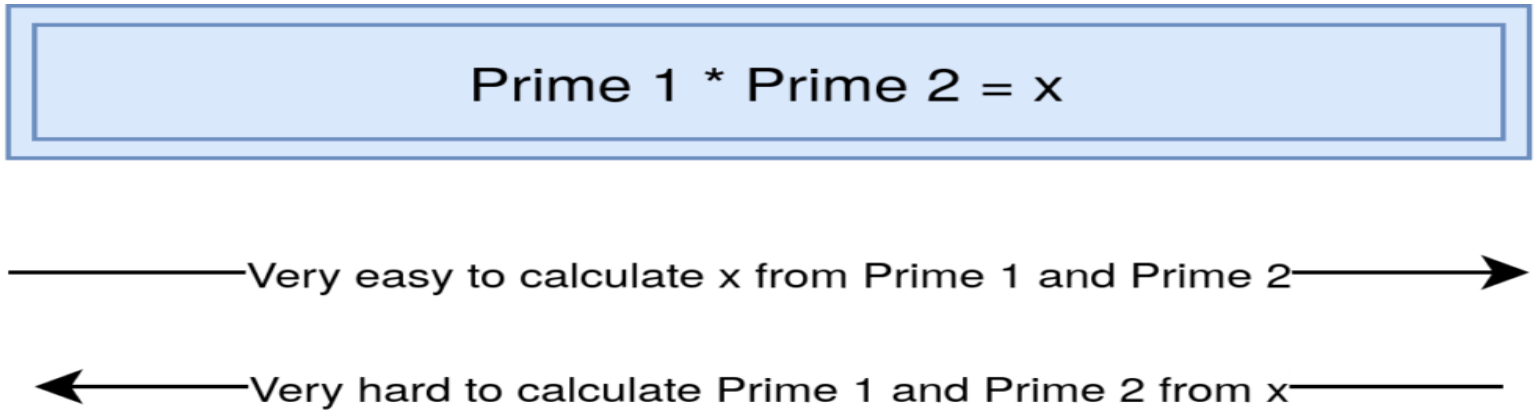
NUMBER THEORY:

- ***Prime number:***
- A number N is called prime number if it has only two factors 1 and N.
- **For example:** 2, 3, 5,7,11, 13, 17, 23 etc.
- Every number can be factorized into its prime numbers.
- **For example:**

Numbers	10	11	100	37	308	14688
Prime Factorization	$2^1 \times 5^1$	$1^1 \times 11^1$	$2^2 \times 5^2$	$1^1 \times 37^1$	$2^2 \times 7^1 \times 11^1$	$2^5 \times 3^3 \times 17^1$
Prime Numbers	2, 5	1, 11	2, 5	1, 37	2, 7, 11	2, 3, 17

- It is very difficult to find the prime factors of a large number. On the other hand, it's very easy to calculate a number with already given primes:
- For example, given prime factors 5 and 7 then we can easily find the natural number from these prime factors as:
- N = 5*7 = 35

- But reverse operation is difficult when number become larger digits.



- We use prime number to generate private and public key for encryption and decryption.

Understanding GCD – Example 1

	12	33
Divisors	1, 2, 3, 4, 6, 12	1, 3, 11, 33
Common Divisors	1, 3	
Greatest Common Divisor (GCD)	3	

∴ GCD(12, 33) = 3

Understanding GCD – Example 2

	25	150
Divisors	1, 5, 25	1, 2, 3, 5, 6, 10, 15, 25, 30, 50, 75, 150
Common Divisors	1, 5, 25	
Greatest Common Divisor (GCD)	25	

∴ GCD(25, 150) = 25

- ***Relatively Prime (Co-prime) Number:***

- Two numbers are said to be Co-prime, if they have no common prime factor except 1.
- OR, we can say, any two numbers whose common greatest prime factor is greater than 1 then, they are not a co-prime number.
- If GCD (a,b) = 1, then a and b are Co-prime numbers.
- GCD = Greatest Common Deviser (Highest Common Factor)

- **Example 1:** Are 4 and 13 Co-prime number?

	4	13
Divisors	1, 2, 4	1, 13
Common Divisors	1	
Greatest Common Divisor (GCD)	1	

- GCD(4,13) = 1
- Hence, they are Co-prime numbers.

- **Example 2:** Are 15 and 21 Co-prime number?

	15	21
Divisors	1, 3, 5, 15	1, 3, 7, 21
Common Divisors	1, 3	
Greatest Common Divisor (GCD)	3	

- GCD (15, 21) = 3
- Here, GCD of 15 and 21 is greater than 1, so they are not a co-prime number.

EUCLIDIAN THEOREM:

- The Euclidean Theorem is a way to find the greatest common divisor of two positive integers. GCD of two numbers is the largest number that divides both of them.

- ***Algorithm:***

1. Take two number a and b
2. Perform a mod b (a should be greater than b)
3. Find new value of a as previous value of b and new value of b as previous value of remainder form a mod b
4. Repeat 2 and 3 until b become zero
5. When value of b is zero, then corresponding value of a is GCD.

Euclidean algorithm to find GCD:

$r_1 = a;$
$r_2 = b;$
$while(r_2 > 0)$
{
$q = r_1/r_2;$
$r = r_1 - q * r_2;$
$r_2 = r;$
$r_1 = r_2;$
}
$GCD(a,b) = r_1$

- **Example 1:** Find the GCD of 12 and 33 using Euclidean Algorithm
- **Solution:** GCD (33, 12) can be calculated form from following table
- 33 is greater than 12 so A=33 and B=12

Q	A	B	R
2	33	12	9
1	12	9	3
3	9	3	0
X	3	0	X

Here, Q is quotient and R is remainder.

The value of a is 3 when value of b is 0.

Hence, GCD(33, 12) = 3.

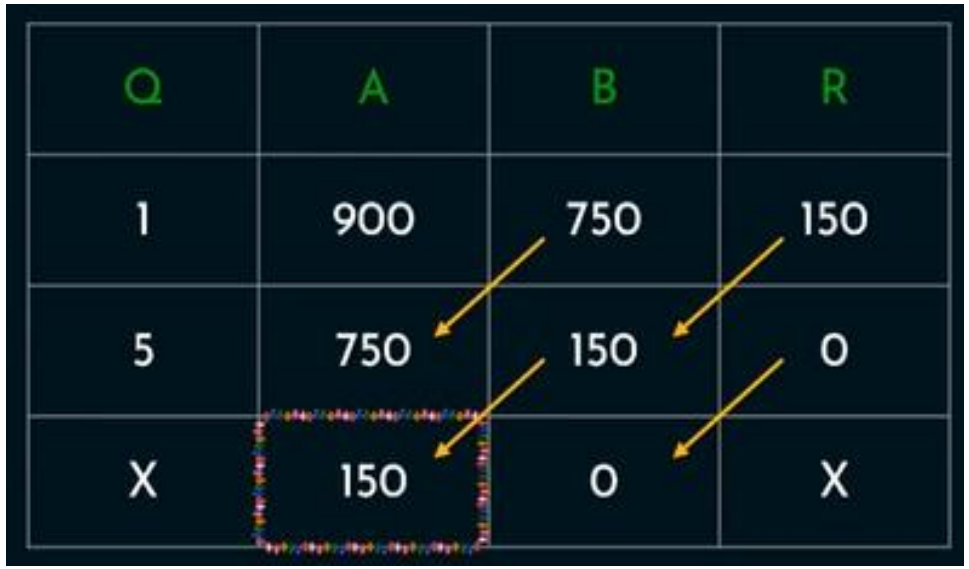
This shows that 33 and 12 are not a co-prime number.

Note:

First perform 33 mod 12

Here, 2 is quotient and 9 is remainder. Then repeat this operation up to b become zero.

- **Example 2:** Find GCD (750, 900)



Q	A	B	R
1	900	750	150
5	750	150	0
X	150	0	X

- Hence, GCD (750, 900) = 150
- This shows that 750 and 900 are not a co-prime number.

Q. $GCD(161, 28) = ?$

Sol^n:

Here,

$a = 161$, $b = 28$

Now,

r_1	r_2	q	r
161	28	5	21
28	21	1	7
21	7	3	0
7	0		

$\therefore GCD(161, 28) = 7$

- GCD(161,28)
- Ans: 7
- GCD (60,25)
- Ans: 5

Set of Residues

- denoted as Z_n.
- is a set of remainders when divided by n i.e. $Z_n = \{0, 1, 2, 3, \ldots\ldots\ldots\ldots, n-1\}$.

E.g. $Z_5 = \{0, 1, 2, 3, 4\}$

$Z_2 = \{0, 1\}$

Operations of Z_n

- Addition, subtraction, multiplication

1. Add 7 to 14 in Z_{15}.

$$(7 + 14) \bmod 15 = 21 \bmod 15 = 6$$

2. Subtract 11 from 7 in Z_{13}.

$$(7 - 11) \bmod 13 = -4 \bmod 13 = 9$$

3. Multiply 11 by 7 in Z_{20}.

$$(11 * 7) \bmod 20 = 77 \bmod 20 = 17$$

Properties of Modular arithmetic for integers in Z_n

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

Residue Class

- denoted as $[a]$ *or* $[a]_n$.
- is the set of integers, when divided by n, we get remainder a.
- i.e. $x \in [a]_n$; $x \equiv a(mod\ n)$

E.g.

Let $n = 4$

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

- All congruence or residue classes of integer modulo 4 are:
- [0] = r € z : r = 0 mod 4

  = r € z : r is divisible by 4 or r=4k for k value can be obtained from 0,-1,1,2,-2 and so on

  = {……,-8,-4,0,4,8,12……..}
- [1] = r € z : r = 1 mod 4

  = r € z : (r – 1) mod 4

  = r € z : r -1 = 4k, r = 4k+1 for k value can be obtained from 0,-1,1,2,-2 and so on

  = = {……,-11,-7,-3,1,5,9,13……..}

Do it for [2] and [3]

Quadratic Residue

- Suppose 'p' is an odd prime and 'a' is an integer.
- 'a' is defined to be quadratic residue if $\boldsymbol{y^2 \equiv a (mod\ p)}$, where $y \in Z_p$

E.g.

$p = 7$

$Z_7 = \{1, 2, 3, 4, 5, 6\}$

$1^2 \equiv 1 (mod\ 7)$
$2^2 \equiv 4 (mod\ 7)$
$3^2 \equiv 2 (mod\ 7)$
$4^2 \equiv 2 (mod\ 7)$
$5^2 \equiv 4 (mod\ 7)$
$6^2 \equiv 1 (mod\ 7)$

$\therefore 1, 2, 4\ are\ quadratic\ residue\ modulo\ 7.$

Note: 1^2 = 1 and 1 mod 7 is also 1
2^2= 4 and 4 mod 7 is also 4
3^2=9 (9 mod 7 is 2 and also 2 mod 7 is 2)
4^2 = 16 (16 mod 7 is 2 and also 2 mod 7 is 2)

Additive inverse of Z_n

Let $a, b \in Z_n$, then a is additive inverse of b if $(\boldsymbol{a + b})\ \boldsymbol{mod\ n = 0}$.

E.g.

1. Find all the additive inverse pairs in Z_5.

Sol^n:

$Z_5 = \{0, 1, 2, 3, 4\}$

The additive inverse pairs in Z_5 are:

(0, 0), *since (0+0) mod 5 = 0.*

(1, 4), *since (1+4) mod 5 =0.*

(2, 3), *since (2+3) mod 5 =0.*

(3, 2), *since (3+2) mod 5 =0.*

(4, 1), *since (4+1) mod 5 =0.*

2. Find the additive inverse of all the elements of Z_{10}.

Sol^{n}:

$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Now,

w	0	1	2	3	4	5	6	7	8	9
$-w$	0	9	8	7	6	5	4	3	2	1

Where, $-w$ is the additive inverse of w.

Multiplicative inverse in Z_n

Let $a, b \in Z_n$, then a is multiplicative inverse of b if $(\boldsymbol{a * b}) \boldsymbol{mod\ n = 1}$.

E.g.

1. Find all the multiplicative inverse pairs in Z_{10}.

Sol^n:

$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

The multiplicative inverse pairs in Z_{10} are;

(1, 1), *since (1*1) mod 10 = 1.*
(3, 7), *since (3*7) mod 10 = 1.*
(7, 3), *since (7*3) mod 10 = 1.*
(9, 9), *since (9*9) mod 10 = 1.*

2. Find multiplicative inverse of each nonzero element in Z_7.

Sol^n:

$Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$

Now,

w	1	2	3	4	5	6
w^{-1}	1	4	5	2	3	6

Where, w^{-1} is the multiplicative inverse of w.

Co-prime

- Also called ***relatively prime***.
- Two positive numbers a and b are co-prime of each other if and only if $\boldsymbol{GCD(a, b) = 1}$.

E.g. (10, 3), (22, 13)

Note: Let $b \in Z_n$ then the multiplicative inverse of b exist if and only if $GCD(b, n) = 1$ i.e. b *and* n are co-prime.

Q. Find the multiplicative inverse of 2 in Z_6.

Sol^n:

Since 2 and 6 are not co-prime so multiplicative inverse of 2 in Z_6 doesn't exist.

- ***Multiplicative Inverse:***
- Multiplicative inverse of a number ‘n’ is defined as:
- $\mathbf{n * n^{-1} = 1}$
- **Example:** Multiplicative invers of 5 is:
- $5 * 5^{-1} = 1$
- Here, the multiplicative inverse of 5 is 5^{-1}
- This is the general case.
- This concept is not valid in the case of modulus.
- The multiplicative inverse of **a mod n** is defined as:
- $\mathbf{a * a^{-1} \equiv 1 \bmod n}$ (Here, **a** and **n** should be co-prime to each other)

Example: What is the multiplicative inverse of 3 (mod 5)?

Solution:

$3 * x \equiv 1 \bmod 5$

Check for all possible value of x that satisfy the above equation

For x = 1

$(3*1) \bmod 5 \equiv 3$ not satisfy

For x = 2

$(3*2) \bmod 5 \equiv 1$This satisfy the above equation.

So, multiplicative inverse of **3 (mod 5) is 2.**

Example 2: What is the multiplicative inverse of 2 (mod 11)?

$2 * x \equiv 1 \bmod 11$

$2 * 6 \equiv 1 \bmod 11$

So, multiplicative inverse of **2 (mod 11) is 6.**

Example 3: What is the multiplicative invers of 2 (mod 10)?

$2 * x \equiv 1 \bmod 10$

Here, we can't find the multiplicative inverse of 2 (mod 10) because 2 and 10 are not co-prime to each other.

- To be co-prime GCD should not be greater than 1. But here GCD of 2 and 10 is 2 which is greater than 1.
- This manual method is suitable for small numbers, but when we need to find the multiplicative inverse of large number it will take so much time to calculate.
- To overcome this problem we need an algorithm, Extended Euclidian Theorem comes here to deal with this problem

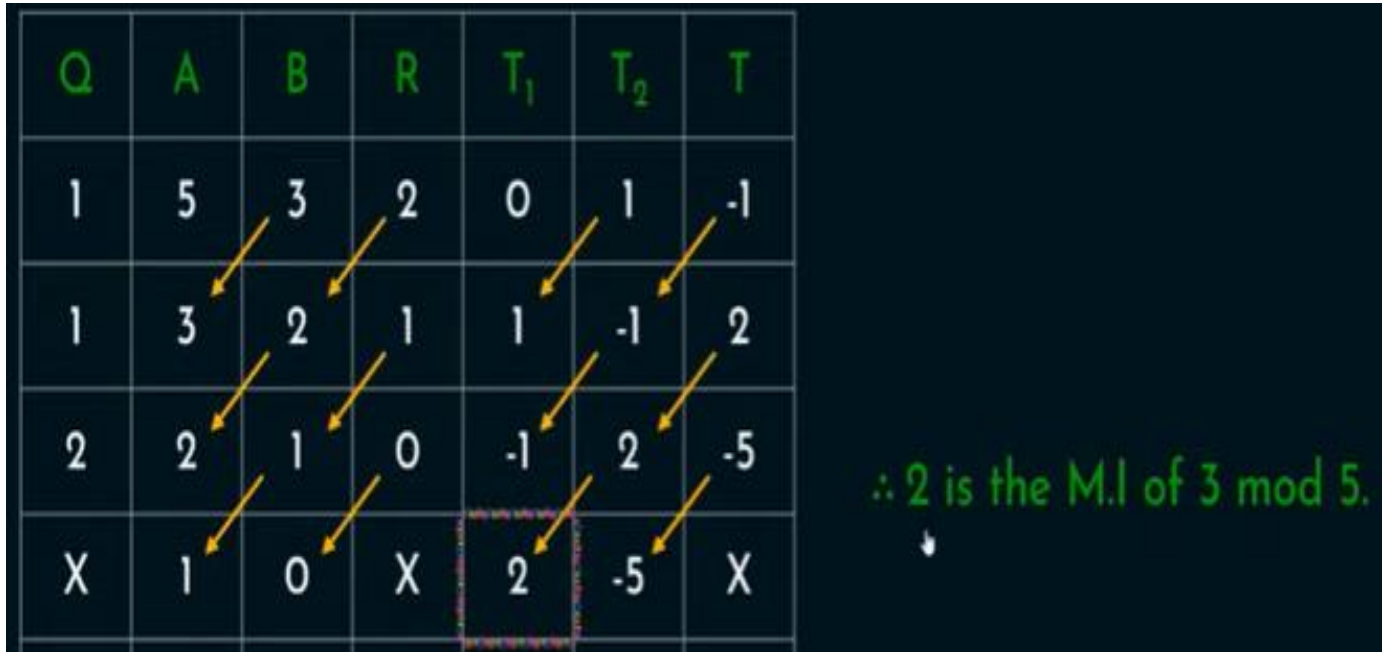
EXTENDED EUCLIDIAN THEOREM:

- Euclidian algorithm is used to find the GCD of any two numbers **a** and **b**. Here, we extended this algorithm to find the multiplicative inverse of **a (mod n).**
- Let's take the extended table of Euclidian Theorem for

Q	**A**	**B**	**R**	**T_1**	**T_2**	**T**

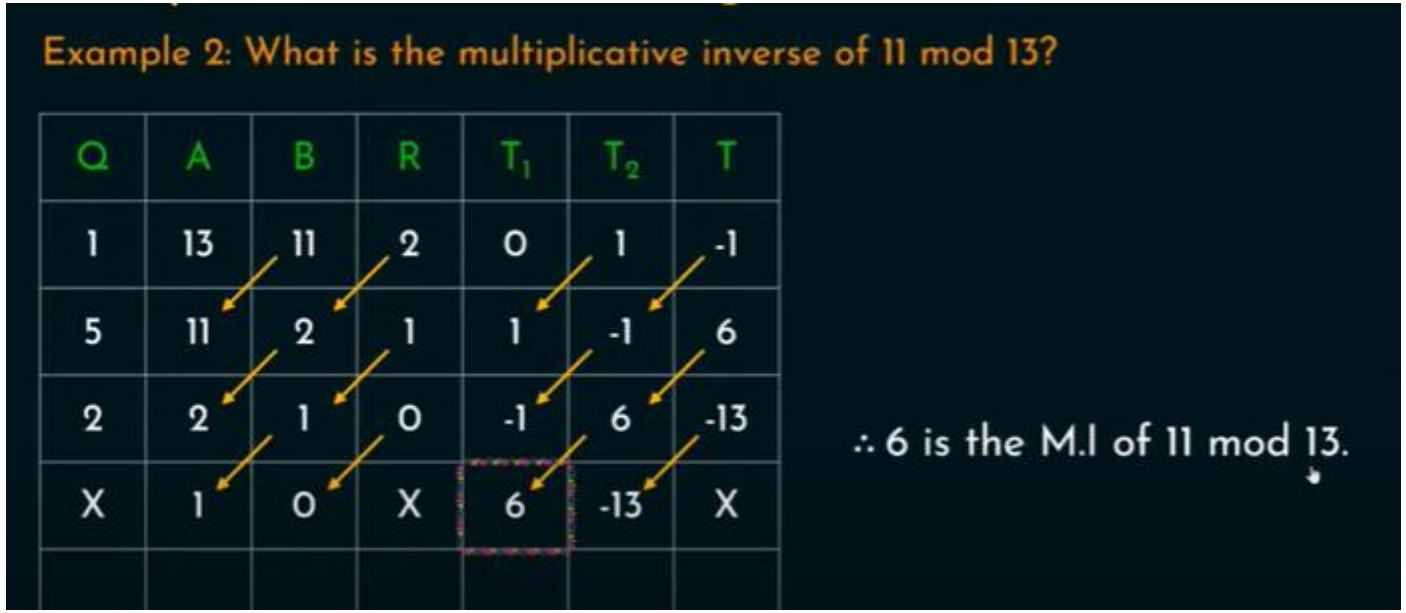
- **Step 1:** Take two numbers A and B
- **Step 2:** Make sure A > B
- **Step 3:** Perform A mod B
- **Step 4:** Place initial value of T_1 and T_2 and 0 and 1 respectively
- **Step 5:** Calculate $T = T_1 – (T_2*Q)$
- **Step 6:** Repeat **Step 3** to **Step 5** until A mod B is possible or until B become zero
- When, B become zero then corresponding value of $\mathbf{T_1}$ is the multiplicative inverse.

- **Example:** Find multiplicative inverse of 3 (mod 5)
- Here, 5 > 3 So, A=5 and B=3



Note: Also you can check 2*3=6 so 6 mod 5 =1 .
To calculate the multiplicative inverse should be equal to 1

- **Example:** Find multiplicative inverse of 11 (mod 13)
- Here, 13 > 11 So, A=13 and B=11



Note: Also you can check 11*6=66 so 66 mod 13 =1 .
To calculate the multiplicative inverse should be equal to 1

E.g.

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

Sol^n:

Here, $b = 11, Z_n = Z_{26}\ i.e\ n = 26$

Now,

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

$\therefore 11^{-1} = -7\ mod\ 26 = 19$

Q. Find the multiplicative inverse of 23 in Z_{100}.

Sol^n:

Here, $b = 23, Z_n = Z_{100}\ i.e\ n = 100$

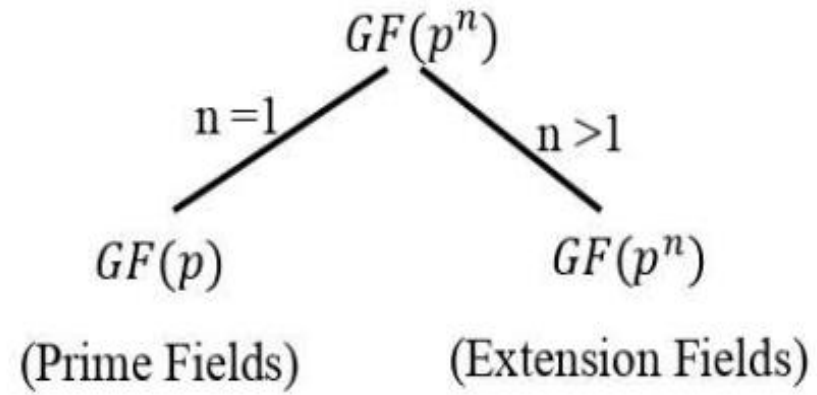
Now,

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	-4
2	23	8	7	1	-4	9
1	8	7	1	-4	9	-13
7	7	1	0	9	-13	100
	1	0		-13	100	

$\therefore 23^{-1} = -13\ mod\ 100 = 87$

Galois Field (GF)

- A Galois field can be defined as a set of numbers that we can add, subtract, multiply and divide together and only ever end up with a result that exists in our set of numbers.
- The number of elements of a Galois field is of the form p^n, where p is a prime and n is a positive integer. Generally, it is denoted by $GF(p^n)$.



For a given prime p, $GF(p)$ is defined as the set $Z_p = \{0, 1, 2, \ldots\ldots\ldots, p-1\}$ of integers together with arithmetic operations modulo p.

Finite Fields of Order p

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

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

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

w	$-w$	w^{-1}
0	0	–
1	1	1

Inverses

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

E.g. Arithmetic in $GF(7)$

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

(a) Addition modulo 7

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

(c) Additive and multiplicative inverses modulo 7

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

(b) Multiplication modulo 7

ARITHMETIC MODULO 8

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

(a) Addition modulo 8

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

(b) Multiplicative modulo 8

w	−w	w^{-1}
0	0	.....
1	7	1
2	6	.....
3	5	3
4	4	.....
5	3	5
6	2	.....
7	1	7

(c) Additive and multiplicative inverse modulo 8

Examples of Finite Field:

- GF(2) = Integer (mod 2) = 5 mod 2 is a finite field
- GF(11) = Integer (mod 11) = 6 mod 11 is a finite field
- GF(12) = Integer (mod 12) is not a finite field because 12 is not a prime number.

Prime Field:

$GF(p^m)$ is called prime field when m=1 and p is a prime number

Example:

- $GF(2) = GF(2^1)$ is a prime field
- $GF(5) = GF(5^1)$ is a prime field

Extension field:

$GF(p^m)$ is called extension field when m>1 and p is a prime number

Example:

$GF(81) = GF(3^4)$ is an extension filed because 3 is prime number

$GF(256) = GF(2^8)$ is an extension field because 2 is a prime number, this is used in AES, this is nothing but group of 8 bits which can make 256 different combinations. The elements in this field are polynomials.

POLYNOMIAL ARITHMETIC

- ***What is Polynomial?***

- The word polynomial is derived from the greek words ‘Poly’ means ‘Many’ and ‘nominal’ means ‘terms’ so combinely it is a ‘many terms’.
- The terms have coefficient and variable raised to the power. These terms are connected by addition or subtraction.
  - **For example:** $3x^2 + 2x - 3$
  - Where, $3x^2$ is one term, 2x is another term, -3 is another term like this it can have any numbers of terms in a polynomial equation.
- A polynomial can have many numbers of terms but not infinite.

In general, polynomial is an expression of the form

$$a_n x^n + a_{n-1} x^{n-1} + \cdots \ldots \ldots + a_1 + a_0$$

for some non-negative integer n and where the coefficient a_0, a_1,……., a_n are drawn from some designated set S. S is called the coefficient set.

- ***Polynomial Arithmetic:***
- Polynomial arithmetic is the arithmetic operation such as addition, subtraction, multiplication and division of polynomial equation with Galois Field or Finite Field.

- GF(5) means there are 5 terms as (0,1,2,3,4). So, maximum value of GF is n-1.

- Addition
- Subtraction (it is an additive inverse so this will performed by addition)
- Multiplication
- Division

Addition:

$$f(x) = a_2x^2 + a_1x + a_0$$
$$g(x) = b_1x + b_0$$
$$f(x) + g(x) = a_2x^2 + (a_1 + b_1)x + (a_0 + b_0)$$

Subtraction:

$$f(x) = a_2x^2 + a_1x + a_0$$
$$g(x) = b_3x^3 + b_0$$
$$f(x) - g(x) = -b_3x^3 + a_2x^2 + a_1x + (a_0 - b_0)$$

Multiplication:

$$f(x) = a_2x^2 + a_1x + a_0$$
$$g(x) = b_1x + b_0$$
$$f(x) * g(x) = a_2b_1x^3 + (a_2b_0 + a_1b_1)x^2 + (a_1b_0 + a_0b_1)$$

Division:

$$f(x) = a_2x^2 + a_1x + a_0$$
$$g(x) = b_1x + b_0$$
$$\frac{f(x)}{g(x)} = ?$$ (Obtained by long division)

Long division for polynomials consists of the following steps:

a) Arrange both the dividend and the divisor in the descending powers of the variable.
b) Divide the first term of the dividend by the first term of the divisor and write the result as the first term of the quotient.
c) Multiply the divisor with the quotient term just obtained and arrange the result under the dividend so that the same powers of x match up. Subtract the expression just laid out from the dividend.
d) Consider the result of the above subtraction as the new dividend and go back to the first step.

As an example, let $f(x) = x^3 + x^2 + 2$ and $g(x) = x^2 - x + 1$, where S is the set of integers. Then

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$
$$f(x) - g(x) = x^3 + x + 1$$
$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

$$\begin{array}{r} x^3 + x^2 \quad + 2 \\ +\ (x^2 - x + 1) \\ \hline x^3 + 2x^2 - x + 3 \end{array}$$

(a) Addition

$$\begin{array}{r} x^3 + x^2 \quad + 2 \\ -\ (x^2 - x + 1) \\ \hline x^3 \quad + x + 1 \end{array}$$

(b) Subtraction

$$\begin{array}{r} x^3 + x^2 \quad + 2 \\ \times\ (x^2 - x + 1) \\ \hline x^3 + x^2 \quad + 2 \\ -x^4 - x^3 \quad - 2x \\ x^5 + x^4 \quad + 2x^2 \\ \hline x^5 \quad + 3x^2 - 2x + 2 \end{array}$$

(c) Multiplication

$$\begin{array}{r|l} & x + 2 \\ \hline x^2 - x + 1 & x^3 + x^2 \quad + 2 \\ & x^3 - x^2 + x \\ \hline & 2x^2 - x + 2 \\ & 2x^2 - 2x + 2 \\ \hline & x \end{array}$$

(d) Division

Figure: Examples of polynomial arithmetic

Q. Calculate the result of the following if the polynomials are over GF(2).

- $(x^7 + x^5 + x^4 + x^3 + x + 1) + (x^3 + x + 1)$
- $(x^7 + x^5 + x^4 + x^3 + x + 1) - (x^3 + x + 1)$
- $(x^7 + x^5 + x^4 + x^3 + x + 1) \times (x^3 + x + 1)$
- $(x^7 + x^5 + x^4 + x^3 + x + 1)/(x^3 + x + 1)$

Sol^n:

Note: Suppose the value after doing any operation is 2 then don't take the value but if its 3 then it can be divided as (2+1) then then 2 value is not taken and only value of 1 is taken

$$
\begin{array}{llll}
x^7 & +x^5+x^4+x^3 & +x+1 \\
 & & +(x^3 & +x+1) \\
\hline
x^7 & +x^5+x^4 &
\end{array}
$$

(a) Addition

$$
\begin{array}{llll}
x^7 & +x^5+x^4+x^3 & +x+1 \\
 & & -(x^3 & +x+1) \\
\hline
x^7 & +x^5+x^4 &
\end{array}
$$

(b) Subtraction

$$
\begin{array}{llll}
 & & x^7 & +x^5+x^4+x^3 & +x+1 \\
 & & & \times (x^3 & +x+1) \\
\hline
 & & x^7 & +x^5+x^4+x^3 & +x+1 \\
 & x^8 & +x^6+x^5+x^4 & +x^2+x & \\
x^{10} & +x^8+x^7+x^6 & +x^4+x^3 & & \\
\hline
x^{10} & & +x^4 & +x^2 & +1
\end{array}
$$

(c) Multiplication

$$
\begin{array}{r|l}
 & x^4+1 \\
\cline{2-2}
x^3+x+1 \;\big/ & x^7+x^5+x^4+x^3+x+1 \\
 & x^7+x^5+x^4 \\
\cline{2-2}
 & x^3+x+1 \\
 & x^3+x+1 \\
\cline{2-2}
 & \times
\end{array}
$$

(d) Division

- **Example 1:** Perform addition, subtraction, multiplication, and division operation on two polynomial equations over GF(2)
- $f(x) = x^4 + x^2 + x + 1$
- $g(x) = x^3 + x + 1$

Solution:

For Addition:

Here, GF(2) = {0,1} this means we can't make coefficient greater than 1. If coefficient is become greater than 1 during addition then perform (mod 2) on that coefficient.

Now,

$f(x) + g(x) = x^4 + x^3 + x^2 + 2x + 2$

Here, 2x + 2 have coefficient greater than 1 so perform (mod 2) on these,

2 mod 2 = 0,

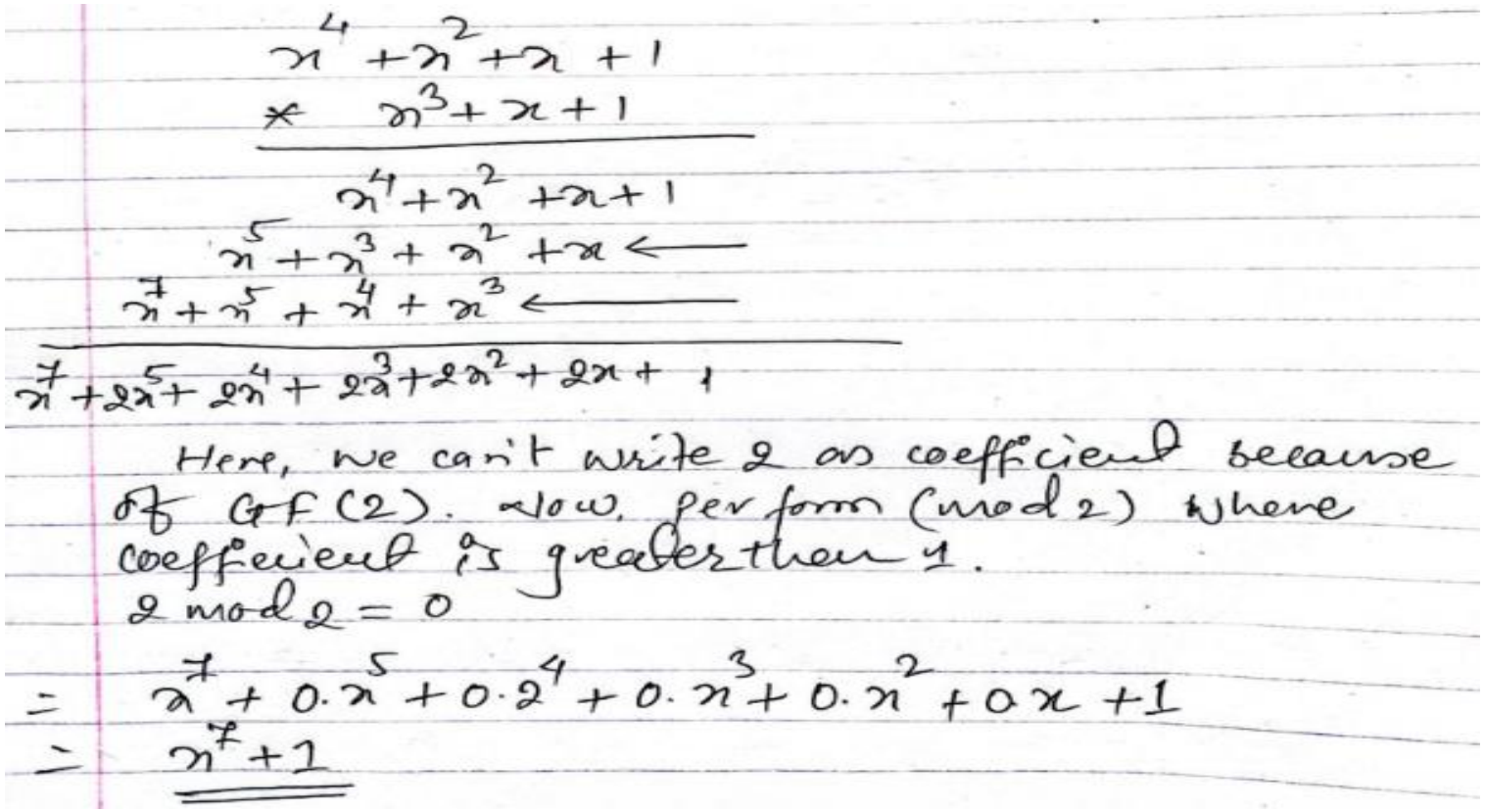
$= x^4 + x^3 + x^2 + 0x + 0$

$= \mathbf{x^4 + x^3 + x^2}$

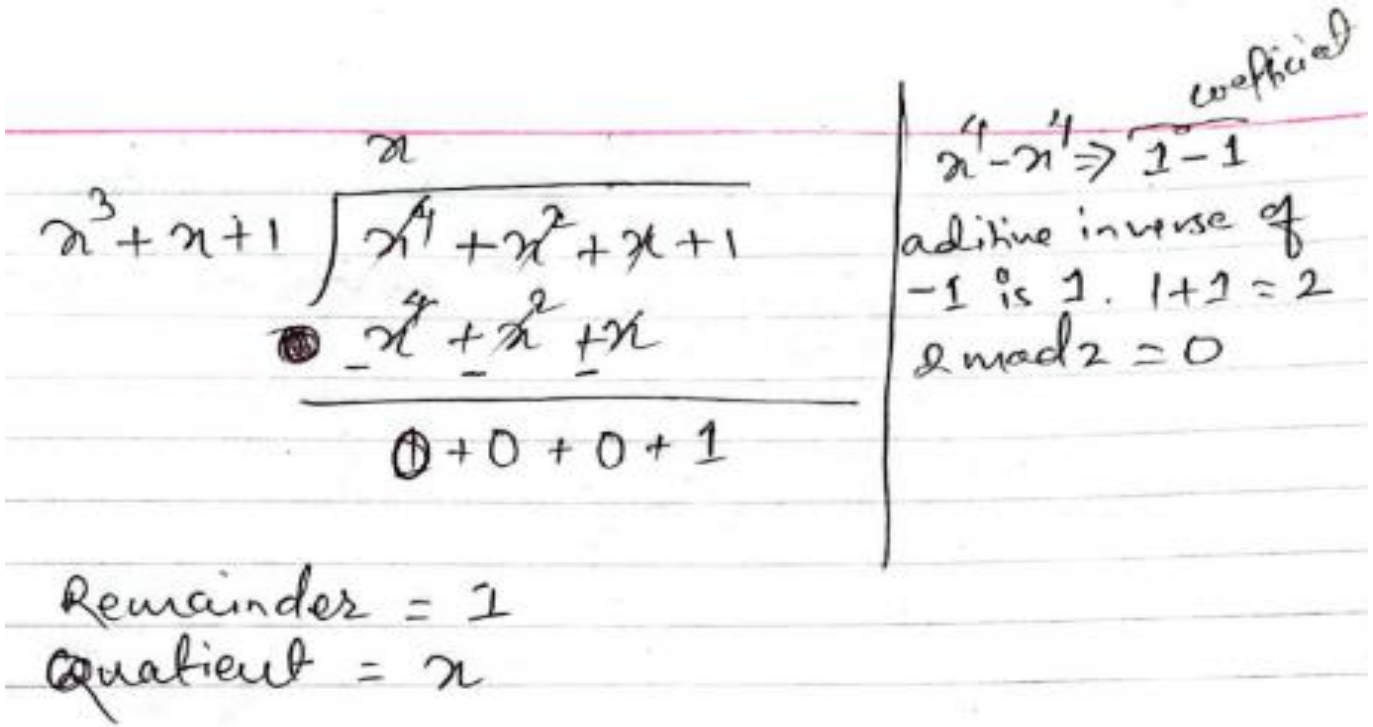
- ***For Subtraction:***
- We perform subtraction operation by additions using additive inverse
- ***Additive Inverse:***
- To find the additive inverse of ‘-a’ with GF(n) then it should satisfy the following equation:
- (a+x) mod n = 0
- Where, x is the any number with GF(n).
- **Example:** Find the additive inverse of -5 with GF(7)
- GF(7) = {0,1,2,3,4,5,6}
- (5+x) mod 7 = 0
- **For, x =0**
- (5+0) mod 7 = 5 not satisfy
- **For, x=1**
- (5+1) mod 7 = 6 not satisfy

- **For, x = 2**
- (5+2) mod 7 = 0 satisfied
- Hence, additive inverse of -5 with GF(7) is 2.
- **New, Subtraction:**
- Perform f(x)-g(x)
- $f(x) – g(x) = x^4 + x^2 + x + 1 – (x^3 + x + 1)$
- $= x^4 + x^2 + x + 1 – x^3 - x - 1$
- Now, find additive inverse of coefficient of negative terms i.e. $– x^3 - x – 1$
- Coefficients of negative terms are, -1, -1, -1
- To find the **Additive Inverse,** find the value from GF(2) = {0,1} and add to 1 and perform (mod 2) remainder should be zero.
- (1+1) mod 2 = 0
- So, additive inverse of -1 is 1, now replace -1 by 1, then question become
- $= x^4 + x^2 + x + 1 + x^3 + x + 1$
- $= x^4 + x^3 + x^2 + 2x + 2$
- $= \mathbf{x^4 + x^3 + x^2}$

- ***For Multiplication:***
- $f(x) * g(x) = (x^4 + x^2 + x + 1) * (x^3 + x + 1)$



- ***For Division:***
- $f(x) / g(x) = (x^4 + x^2 + x + 1) / (x^3 + x + 1)$



Q. Divide $5x^2 + 4x + 6$ by $2x + 1$ over GF(7).

Sol^n:

$$
\begin{array}{r}
6x + 6 \\
2x + 1 \overline{)\,5x^2 + 4x + 6} \\
\cancel{12x^2 + 6x} \\
\underline{\;5x^2 + 6x\;} \\
\cancel{-2x + 6} \\
5x + 6 \\
\cancel{12x + 6} \\
\underline{\;5x + 6\;} \\
\times
\end{array}
$$

1) We must start by dividing $5x^2$ by $2x$. This requires that we divide 5 by 2 in GF(7). Dividing 5 by 2 is the same as multiplying 5 by the multiplicative inverse of 2. Multiplicative inverse of 2 is 4 since $2 * 4 \; mod \; 7 = 1$. So we have $\frac{5}{2} = 5 * 2^{-1} = 5 * 4 = 20 \; mod \; 7 = 6$. Therefore, the first term of the quotient is $6x$.
2) Since the product of $6x$ and $2x + 1$is $(12x^2 + 6x) \; mod \; 7 = 5x^2 + 6x$, we need to subtract $5x^2 + 6x$ from the dividend $5x^2 + 4x + 6$.The result is $(-2x + 6) \; mod \; 7 = 5x + 6$.
3) Our new dividend for the next round of long division is therefore $5x + 6$. To find the next quotient term, we need to divide $5x$ by the first term of the divisor that is by $2x$. Reasoning as before, we see that the next quotient term is again 6.
4) Applying the same process as above, we get the remainder is zero.

*find the term from GF(7) whose multiplicaton with 2 makes 5 either directly or by (mod7)
(2x6) mod7 = 5

- ***Modular Exponential:***
- It is a type of exponential which is performs over a modulus.

- It is denoted by $\mathbf{a^b \bmod n}$ or $\mathbf{a^b \pmod n}$
- **Example 1:** Solve $23^3 \bmod 30$

$23^3 \bmod 30 = -7^3 \bmod 30$ || 23 mod 30 can be 23 or -7

$= -7^3 \bmod 30$

$= -7^2 \times -7 \bmod 30$

$= 49 \times -7 \bmod 30$

$= -133 \bmod 30$

$= -13 \bmod 30$

$= 17 \bmod 30$

$23^3 \bmod 30 = 17$

- **Example 2:** Solve $31^{500} \bmod 30$

$$
\begin{aligned}
31^{500} \bmod 30 &= 1^{500} \bmod 30 \\
&= 1 \bmod 30 \\
&= 1 \\
31^{500} \bmod 30 &= 1
\end{aligned}
$$

- **Example 3:** Solve $242^{329} \bmod 243$

$$
\begin{aligned}
242^{329} \bmod 243 &= -1^{329} \bmod 243 \\
&= -1^{329} \bmod 243 \;||\; -1^{328} \times -1^{1} \\
&= -1 \bmod 243 \\
&= 242 \\
242^{329} \bmod 243 &= 242
\end{aligned}
$$

- **Example 4:** Solve 11^7 mod 13

$$
\begin{aligned}
11^7 \bmod 13 &= 11 \bmod 13 \times 11 \bmod 13 \times 11 \bmod 13 \times 11 \bmod 13 \times 11 \bmod 13 \times 11 \bmod 13 \times 11 \bmod 13 \\
&= -2 \times -2 \times -2 \times -2 \times -2 \times -2 \times -2 \bmod 13 \\
&= -128 \bmod 13 \\
&= -11 \bmod 13 \\
&= 2 \\
11^7 \bmod 13 &= 2
\end{aligned}
$$

- **Example 5:** Solve 88^7 mod 187
- Perform modulus operation as power 1, 2, 4, 8, 16,……..

$$
\begin{aligned}
88^1 \bmod 187 &= 88 \\
88^2 \bmod 187 &= 88^1 \times 88^1 \bmod 187 = 88 \times 88 = 7744 \bmod 187 = 77 \\
88^4 \bmod 187 &= 88^2 \times 88^2 \bmod 187 = 77 \times 77 = 5929 \bmod 187 = 132 \\
88^7 \bmod 187 &= 88^4 \times 88^2 \times 88^1 \bmod 187 = (132 \times 77 \times 88) \bmod 187 \\
&= 894{,}432 \bmod 187 \\
88^7 \bmod 187 &= 11
\end{aligned}
$$

FERMET'S THEOREM:

- If '**p**' is a prime number and '**a**' is a positive integer not divisible by '**p**' then,

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

- This is we can say a^{p-1} is congruent to 1 (mod p). This statement means result of
- $(a^{p-1} \bmod p) = 1$.
- **Example 1:** If p = 5 and a = 2, check these two numbers proves the Fermat's theorem or not?

Given: p=5 and a=2.

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

$2^{5-1} \equiv 1 \pmod{5}$

$2^4 \equiv 1 \pmod{5}$

$16 \equiv 1 \pmod{5}$

Therefore, Fermat's theorem holds true for p=5 and a=2.

Note: 16 mod 5 = 1
1 mod 5 =1

- **Example 2:** Prove Fermat's theorem holds true for p = 13 and a = 11.

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

$11^{13-1} \equiv 1 \pmod{13}$

$11^{12} \equiv 1 \pmod{13}$

$-2^{12} \equiv 1 \pmod{13}$

$-2^{4x3} \equiv 1 \pmod{13}$

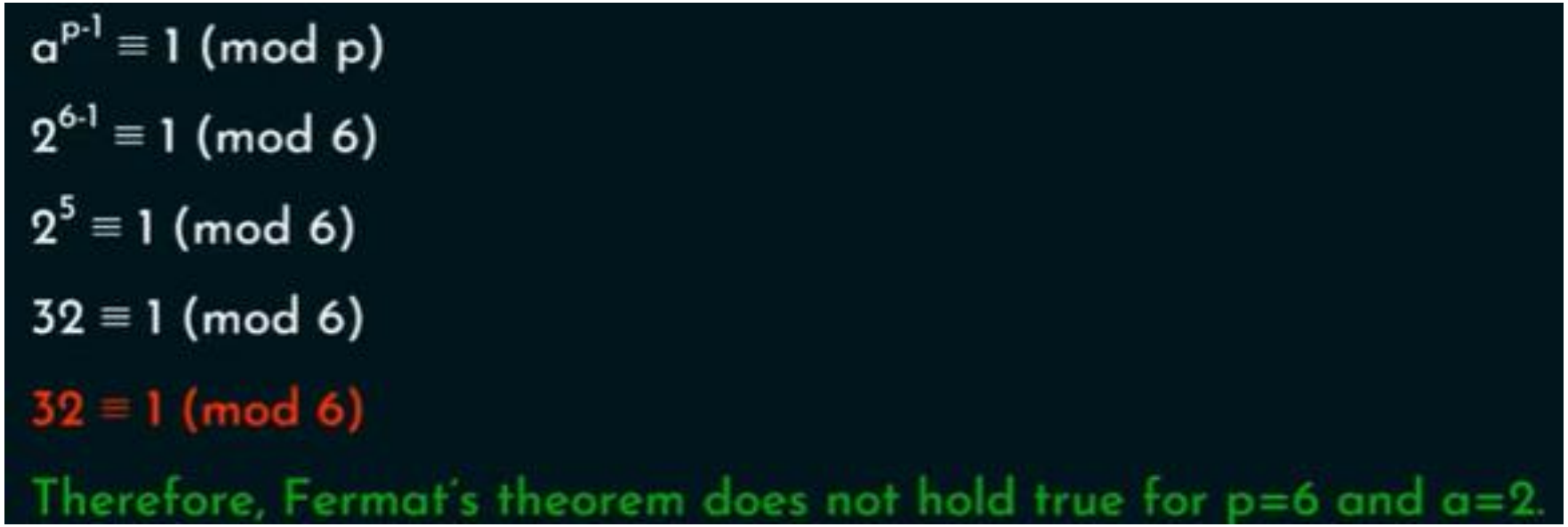
$3^3 \equiv 1 \pmod{13}$

$27 \equiv 1 \pmod{13}$

Therefore, Fermat's theorem holds true for p=13 and a=11.

Note:
In 11^12 use modular exponent
i.e. 11-13= -2
So it can be represent in -2^12
Again -2^4*3 = 16^3
16-13 = 3
So it can be represented in 3^3
27 mod 13 =1
1 mod 13 =1

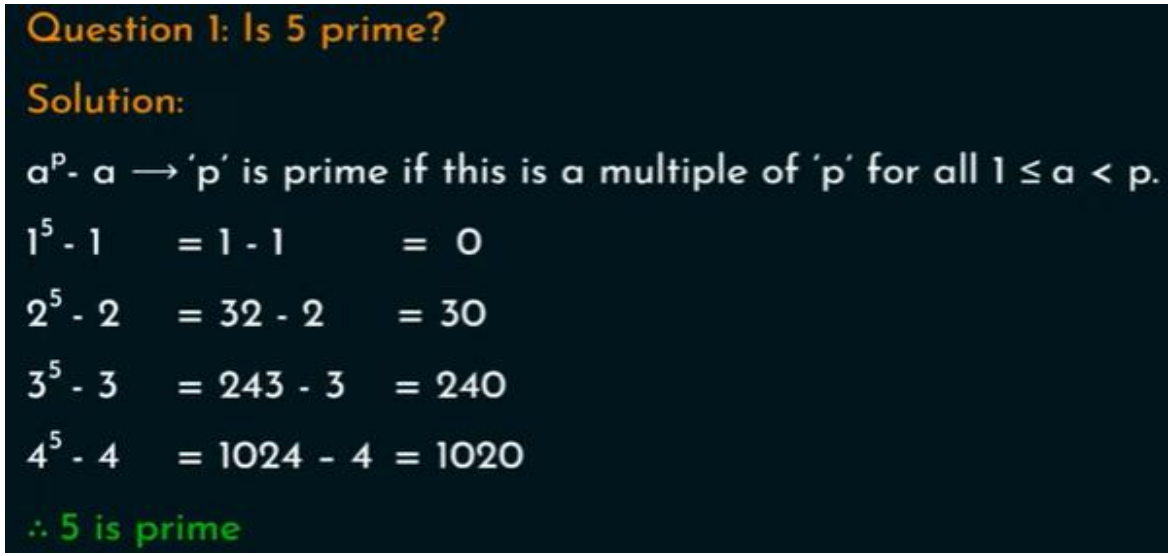
- **Example 3:** Prove Fermat's theorem does not holds true for p = 6 and a = 2



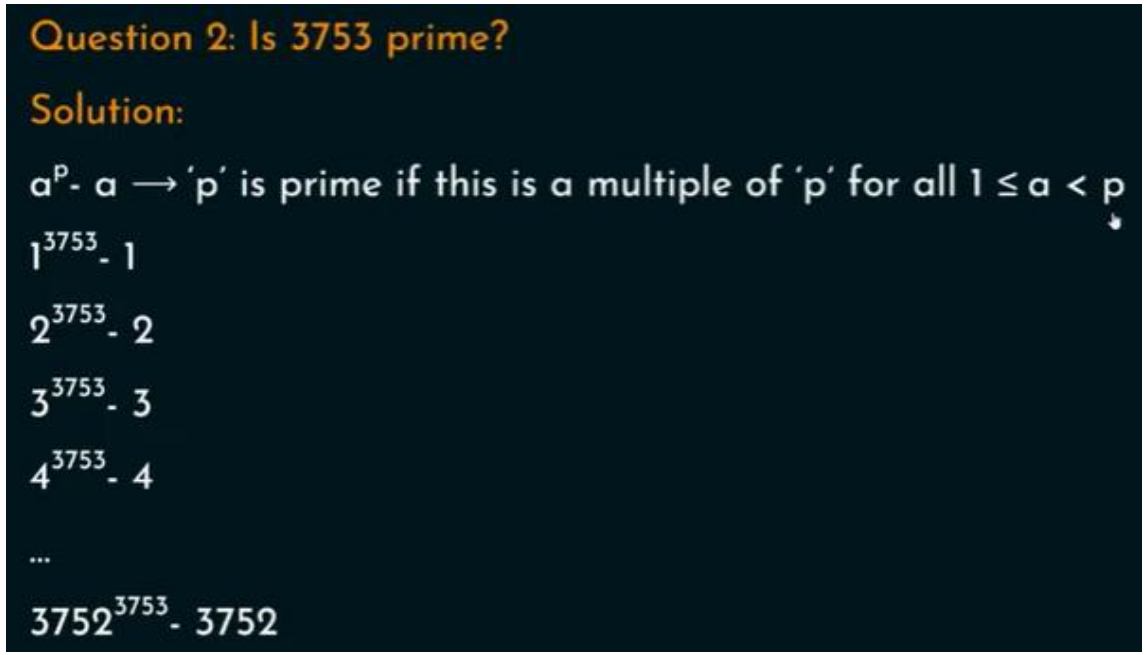
Note: 32 mod 6 = 2 and 1 mod 6 = 1 so for this it does not hold true

PRIMALITY TESTING:

- ***Fermat's Testing Algorithm:***
- Test for $\mathbf{a^p - a}$,
- **'p'** is prime if this is a multiples of '**p**' for all **1<=a<p**
- **Example 1:**



- This algorithm is suitable only for small prime number, if number is large then calculation should repeated so many times which is time consuming.
- **For example:**



- This is the main drawback of Fermat's Algorithm.
- Another drawback is accuracy. Fermat's primality test can't provide 100% accuracy.

MILLER-RABIN ALGORITHM

- Used to test the primality of large numbers.
- To test whether a given number ‘n’ is prime or not, Miller Rabin algorithm works as follows:
- ***Algorithm:***

1. *Write* $n - 1 = 2^k m$, *where* m *is odd.*
2. *Choose a random number* a; $1 \leq a \leq n - 1$.
3. *Compute* $b = a^m \bmod n$
4. *If* $b \equiv 1 (\bmod n)$ *then return PRIME.*
5. *For* $i = 0$ *to* $k - 1$
   do
   IF $b \equiv -1 (\bmod n)$ *then return PRIME.* $b \equiv -1 (\bmod n) \Rightarrow (b+1) \bmod n = 0$
   ELSE $b = b^2 \bmod n$
6. *Return COMPOSITE.*

➢ ***Two properties of prime number given by Rabin-Miller:***

Property 1:

If p is prime and a is a positive integer less than p, then $a^2 \ mod \ p = 1$if and only if either $a \ mod \ p = 1$ or $a \ mod \ p = -1 \ mod \ p = p - 1$.

Property 2:

Let p be a prime number greater than 2. We can then write $p - 1 = 2^q k$, with $k > 0 \ and \ q$ as odd. Let a be any integer such that $1 < a < p - 1$ then one of the following conditions is true:

- $a^q \ mod \ p = 1$
- One of the number $a^q, a^{2q}, a^{4q}, \ldots\ldots\ldots, a^{2^{k-1}q}$ is congruent to 1 $mod \ p$.

Q. Determine whether the integer 17 is prime or not using Miller-Rabin algorithm.

Sol^n:

$n = 17$

$n - 1 = 16 = 2^4 * 1\,,\ \ k = 4\ \&\ m = 1$

$a \rightarrow (1 - 16); a = 5$

$b = a^m \ mod\ n = 5^1\ mod\ 17 = 5$

$b \equiv 1(mod\ n) \Rightarrow 5 \equiv 1(mod\ 17)$ which is false.

So,

$i = 0\ to\ k - 1\ \ i.e.\ i = 0\ to\ 3$

$i = 0$

$b = 5$

$5 + 1 = 6\ mod\ 17 = 6$

$b = b^2\ mod\ n = 25\ mod\ 17 = 8$

$b \equiv -1(mod\ n) \Rightarrow (b + 1)\ mod\ n = 0$

$i = 1$

$b = 8$

$8 + 1 = 9\ mod\ 17 = 9$

$b = b^2\ mod\ n = 64\ mod\ 17 = 13$

$i = 2$

$b = 13$

$13 + 1 = 14\ mod\ 17 = 14$

$b = 13^2\ mod\ 17 = 169\ mod\ 17 = 16$

$i = 3$

$b = 16$

$16 + 1 = 17\ mod\ 17 = 0$

$\therefore$ 17 is prime.

Q. Determine whether the integer 7 is prime or not using Miller-Rabin algorithm.

Sol^{n}:

$n = 7$

$n - 1 = 6 = 2^1 * 3\ ,\ \ k = 1\ \&\ m = 3$

$a \rightarrow (1 - 6); a = 4$

$b = a^m \ mod\ n = 4^3\ mod\ 7 = 64\ mod\ 7 = 1$

$b \equiv 1(mod\ n) \Rightarrow 1 \equiv 1(mod\ 7)$ which is true. So, 7 is prime.

Q. Find if 27 is prime or not using Miller-Rabin Algorithm

EULER TOTIENT FUNCTION

- Euler Totient Function also called **Phi Function** is the set of number of positive integers less than 'N' that are relatively prime or Co-prime to N. It is denoted by symbol Phi (Φ)
- **Example 1:** Find the Φ(5)

Here, N = 5

The numbers less than 5 are: 1, 2, 3, 4

Now, out of these four numbers, fine out which numbers are the co-prime to the 5. We knew co-prime numbers are those whose GCD is 1

GCD	Relatively Prime?
GCD (1, 5) = 1	✓
GCD (2, 5) = 1	✓
GCD (3, 5) = 1	✓
GCD (4, 5) = 1	✓

∴ Φ(5) = 4.

- **Example 2:** Find the Φ(8)
- Here, N = 8
- The numbers less than 8 are: 1, 2, 3, 4, 5, 6, 7
- Now, out of these numbers fine out which numbers are the co-prime to the 8.

GCD	Relatively Prime?
GCD (1, 8) = 1	✓
GCD (2, 8) = 2	✗
GCD (3, 8) = 1	✓
GCD (4, 8) = 4	✗

GCD	Relatively Prime?
GCD (5, 8) = 1	✓
GCD (6, 8) = 2	✗
GCD (7, 8) = 1	✓

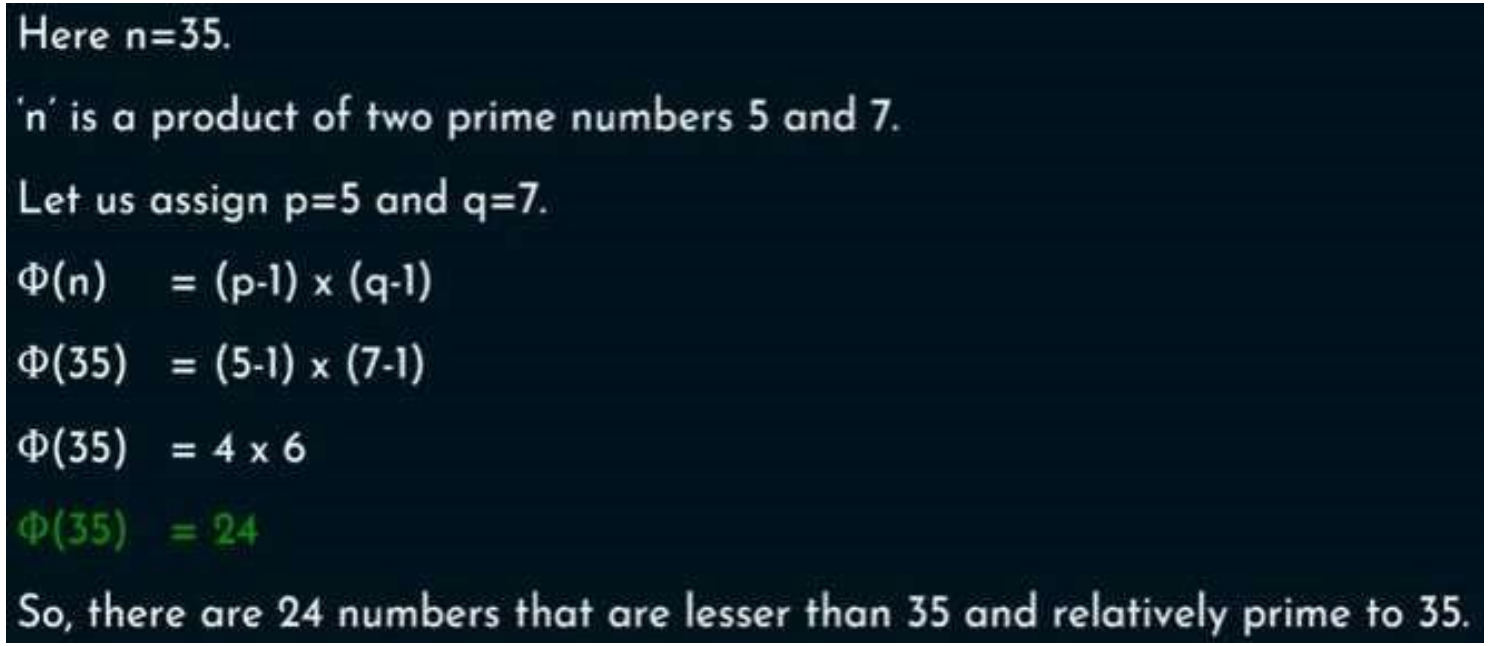
∴ Φ(8) = 4.

- This technique is suitable for small numbers only. If the number is large enough, it too much time consuming to calculate the value of Phi by this manual method.
- To deal with large numbers following formula is used:

$\Phi(n)$	Criteria of 'n'	Formula
	'n' is prime.	$\Phi(n) = (n-1)$
	n = p x q. 'p' and 'q' are primes.	$\Phi(n) = (p-1) \times (q-1)$
	n = a x b. Either 'a' or 'b' is composite. Both 'a' and 'b' are composite.	$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \ldots .$ where $p_1, p_2, \ldots$ are distinct primes.

- In third condition **n = a*b** where **a** and **b** both or one of these are not prime,
- First find the prime factor of **n**, where p1 is first prime factor, p2 is second prime factor and so on. Then calculate totient function as formula.

Example 1: Fine the Φ(35)



Note that, in the product of prime numbers both numbers should be different for this condition.

For example, if we try to find Φ(25)

N = 25, is product of two prime numbers 5 and 5

But here we can‟t apply the formula (p-1)*(q-1) because both the prime are same. To deal this type of problem we should use third formula as mention above.

N = 25, prime factor of 25 is 5^2

So, p1 = 5

Φ(25) = N * (1- 1/p1)

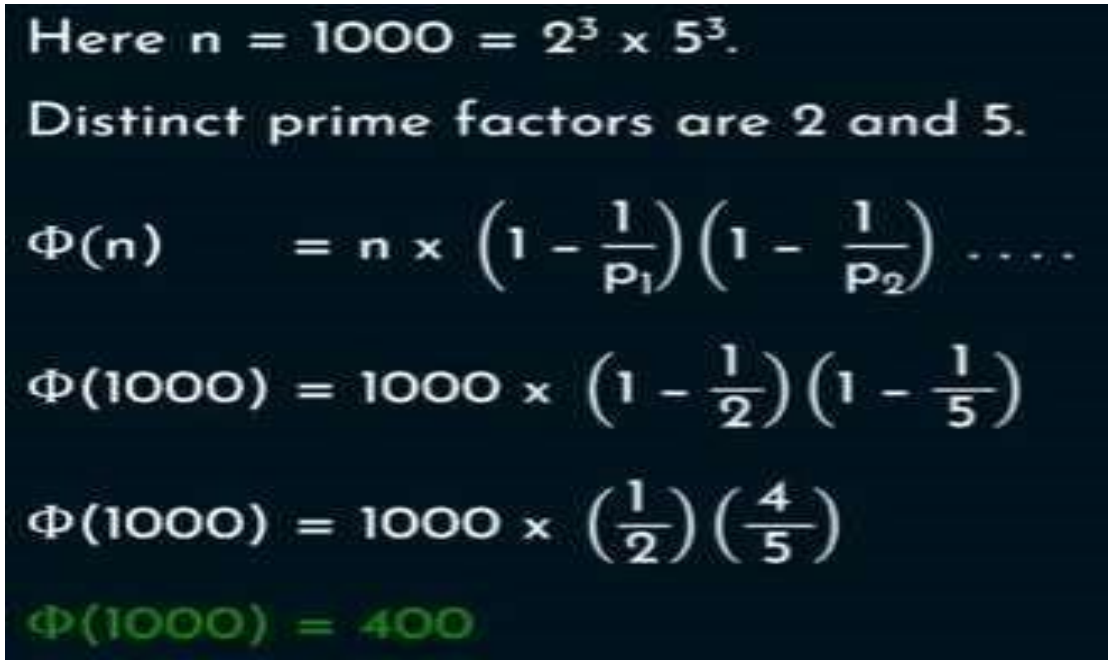
= 25 * (1- 1/5)

= 25 * 4/5

= 20

So, there are 20 numbers that are less than 25 and relatively prime to 25.

- **Example 2:** Find Φ(1000)



- So, there are 400 numbers that are less than 1000 and relatively prime to 1000.

Euler's Theorem

Euler's theorem states that if 'a' and 'n' are co-prime positive integers then,

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

Where, $\phi(n)$ is Euler's totient function.

- **Example 1:** Prove Euler's theorem hold true for a = 3 and n = 10

Given: a=3 and n=10.

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

$3^{\Phi(10)} \equiv 1 \pmod{10}$

$\Phi(10) = 4$

$3^4 \equiv 1 \pmod{10}$

$81 \equiv 1 \pmod{10}$

Therefore, Euler's theorem holds true for a=3 and n=10.

Example 2: Prove Euler's theorem does not hold true for a = 2 and n = 10

Given: a=2 and n=10.

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

$2^{\Phi(10)} \equiv 1 \pmod{10}$

$\Phi(10) = 4$

$2^4 \equiv 1 \pmod{10}$

$16 \equiv 1 \pmod{10}$

Therefore, Euler's theorem does not hold for a=2 and n=10.

ADVANCED ENCRYPTION STANDARDS (AES)

- The Advanced Encryption Standard (AES) is a symmetric block cipher established in 2001 by U.S. NSIT (National Institute of Standards and Technology) and used by U.S. government to protect classified information.

- AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, Cybersecurity and electronic data protection.

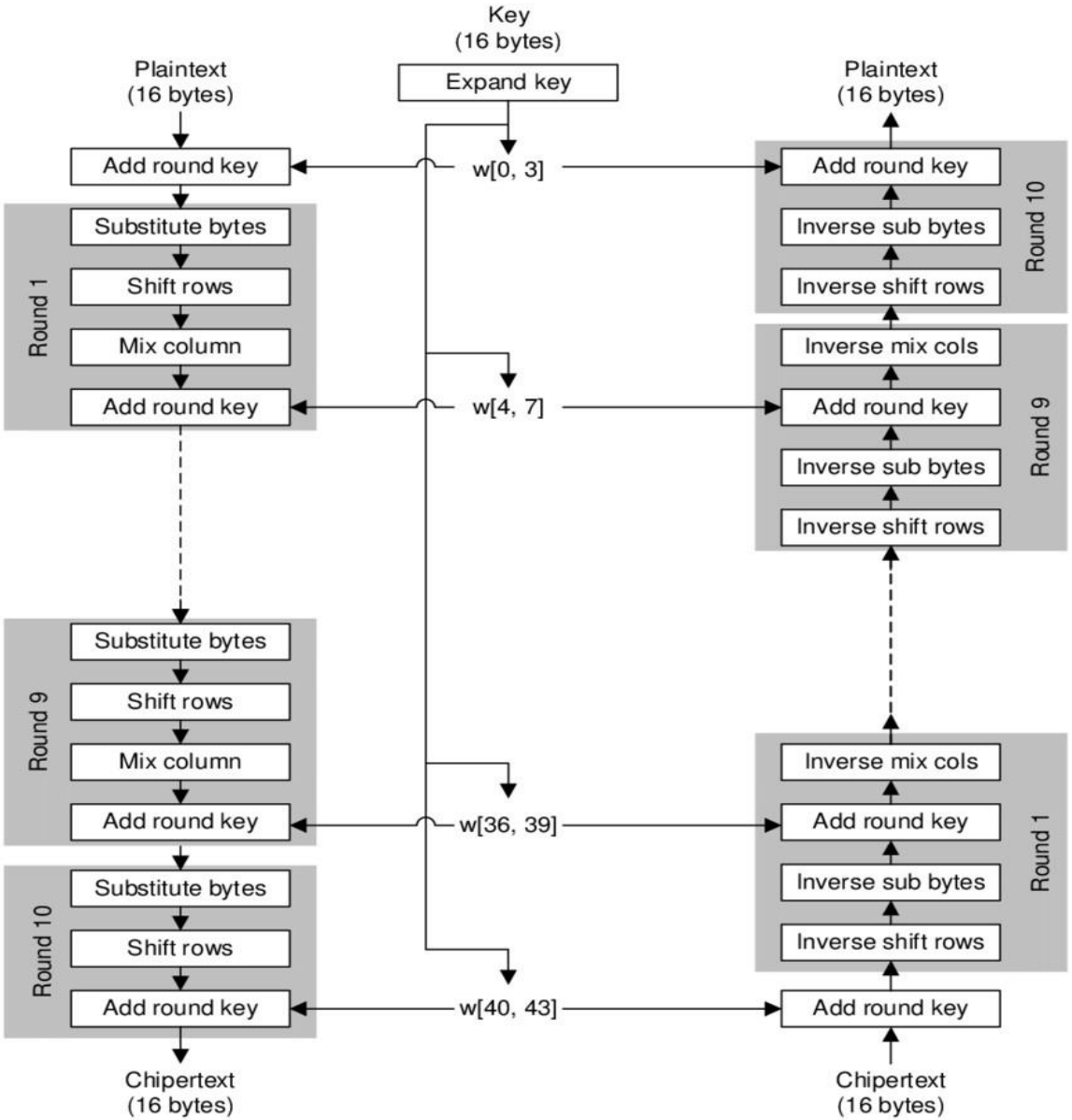
- AES consists of multiple rounds of processing based of different key sizes such as:

  - 128 bit keys – 10 rounds of encryption operation
  - 192 bit keys – 12 rounds of encryption operation
  - 256 bit keys – 14 rounds of encryption operation

- ***Terms to remember:***

- 1 byte = 8 bits

- 1 word = 4 bytes (32 bits)



- Each round consists of four functions:
  1. Sub Bytes
  2. Shift Rows
  3. Mix Columns, not applied in last round.
  4. Add Round Key

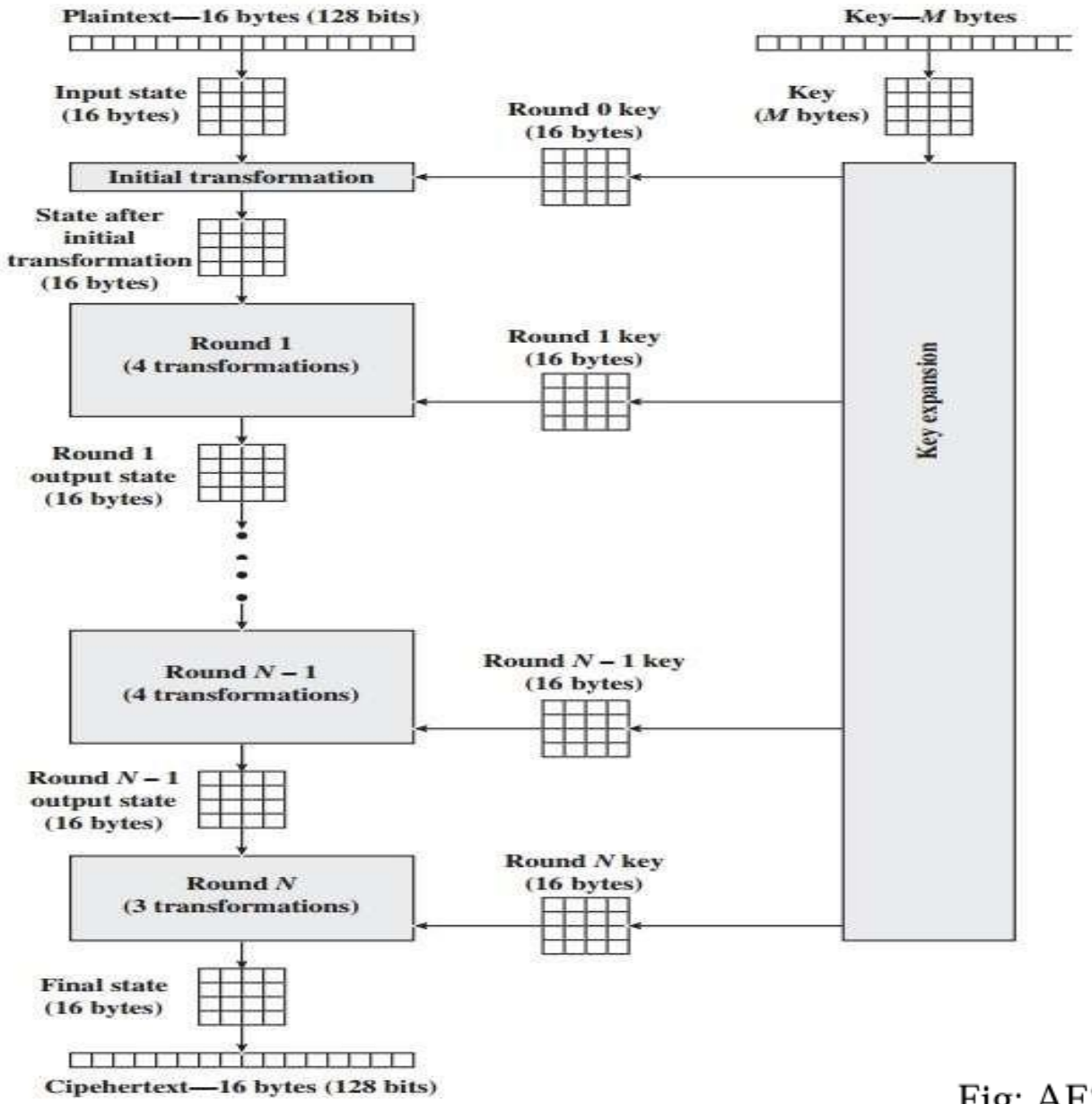


Fig: AES Encryption Process

AES treats the 128-bits block (16 bytes) as a 4×4 byte array, called state matrix.

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{12}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

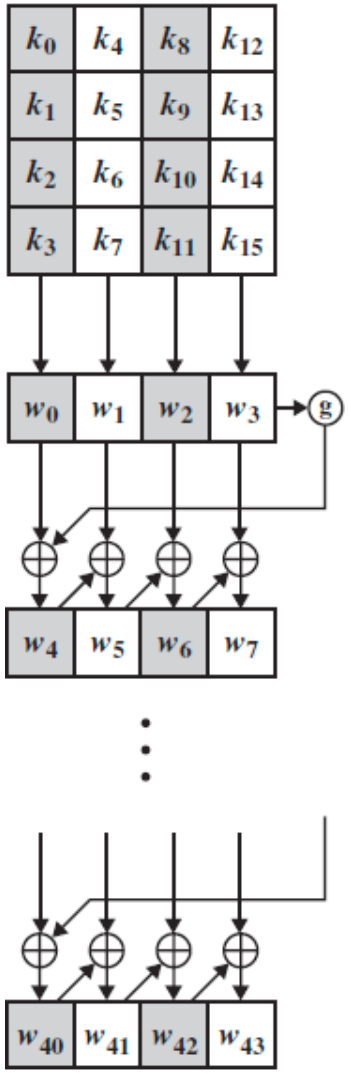
All the four AES operations are applied on the matrices further. These operations can be described as:

KEY EXPANSION

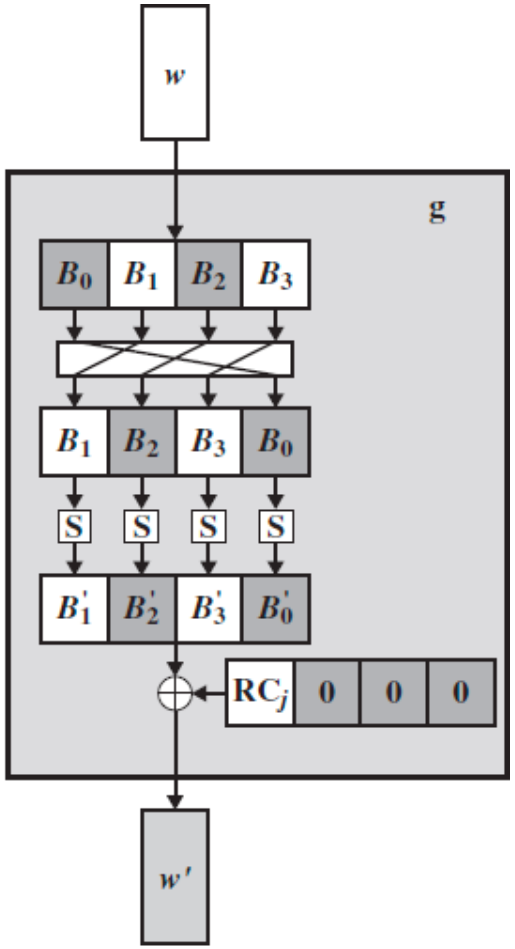
AES Key Expansion

To create round keys for each round, AES uses a key expansion process. If the number of rounds is N_r, the key expansion routines creates $N_r + 1$ 128-bit round keys for one single 128-bit cipher key.
It takes 128 bits (16-bytes) key and expands into array of 44 32-bit words. For the purpose of adding the key to state, each word is considered as column matrix.

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



(a) Overall algorithm



(b) Function g

- The elements k_0 to k_{15} in initial matrix are 16 bytes of 128-bits key.
- The elements B_0 to B_3 in g function is 4 bytes in a word say w_3
- w_0 is a word formed by 4 bytes k_0 k_1 k_2 k_3 likewise other words formed.

Function g performs the following operations:

- **Step 1:** Perform 1 byte circular left shift
- **Step 2:** Using constant S-Box each sub word performs a byte substitution
- **Step 3:** Finally, output of Step 2 is XORed with Round Constant RC_j

Round	Words
Pre-round	w_0 w_1 w_2 w_3
1	w_4 w_5 w_6 w_7
2	w_8 w_9 w_{10} w_{11}
…	…
N_r	w_{4N_r} w_{4N_r+1} w_{4N_r+2} w_{4N_r+3}

Calculation of $\boldsymbol{g}$ function includes following 3 processes:

1. **RotWord** performs a one byte circular left shift on a word. For e.g. $RotWord[b_0, b_1, b_2, b_3] \rightarrow [b_1, b_2, b_3, b_4]$
2. **SubWord** performs a byte substitution on each byte of input word using S-box.
3. The result of step 1 and 2 is XORed with $RCon[j]$- the round constant.

Round Constant (RCon):

- The round constant is a word in which the three rightmost bytes are always 0.
- It is different for each round and defined as:

$$RCon[j] = (RCon[j], 0, 0, 0)$$

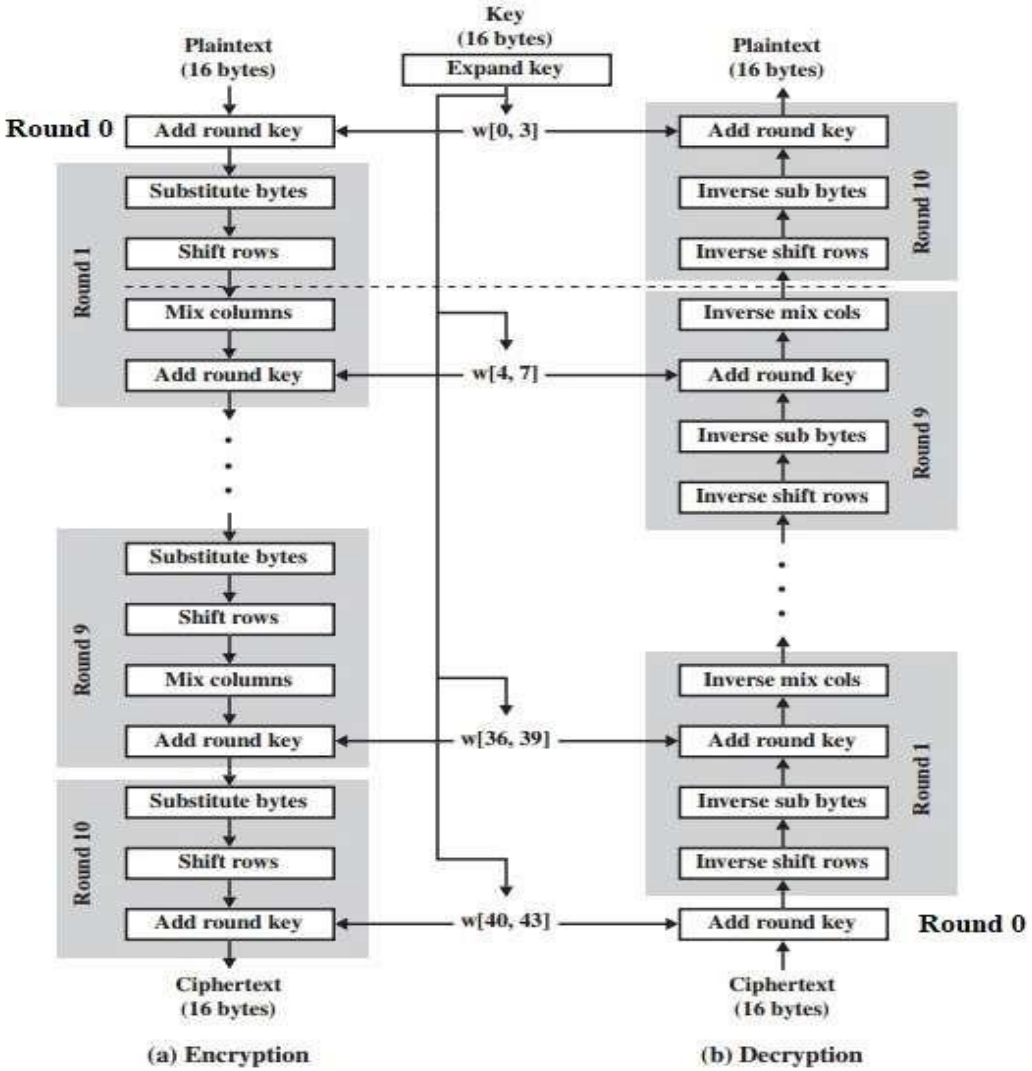
Where, $RCon[1] = 1, RCon[j] = 2 * RCon[j-1]$

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\mathbf{01}\ 00\ 00\ 00)_{16}$	6	$(\mathbf{20}\ 00\ 00\ 00)_{16}$
2	$(\mathbf{02}\ 00\ 00\ 00)_{16}$	7	$(\mathbf{40}\ 00\ 00\ 00)_{16}$
3	$(\mathbf{04}\ 00\ 00\ 00)_{16}$	8	$(\mathbf{80}\ 00\ 00\ 00)_{16}$
4	$(\mathbf{08}\ 00\ 00\ 00)_{16}$	9	$(\mathbf{1B}\ 00\ 00\ 00)_{16}$
5	$(\mathbf{10}\ 00\ 00\ 00)_{16}$	10	$(\mathbf{36}\ 00\ 00\ 00)_{16}$

- We need total number of 11 round keys. Round 0 key for initial transformation i.e. initial AddRoundKey and remaining 10 for Round 1 to Round 10 of encryption process.
- Round keys for 128-bits keys are:

Round	Words
Initial Transformation key (Round 0 key)	w_0 w_1 w_2 w_3 (This is original key)
Round 1 key	w_4 w_5 w_6 w_7
Round 2 key	w_8 w_9 w_{10} w_{11}
……	……
Round 10 key	w_{40} w_{41} w_{42} w_{43}

ENCYRPTION PROCESS OF AES



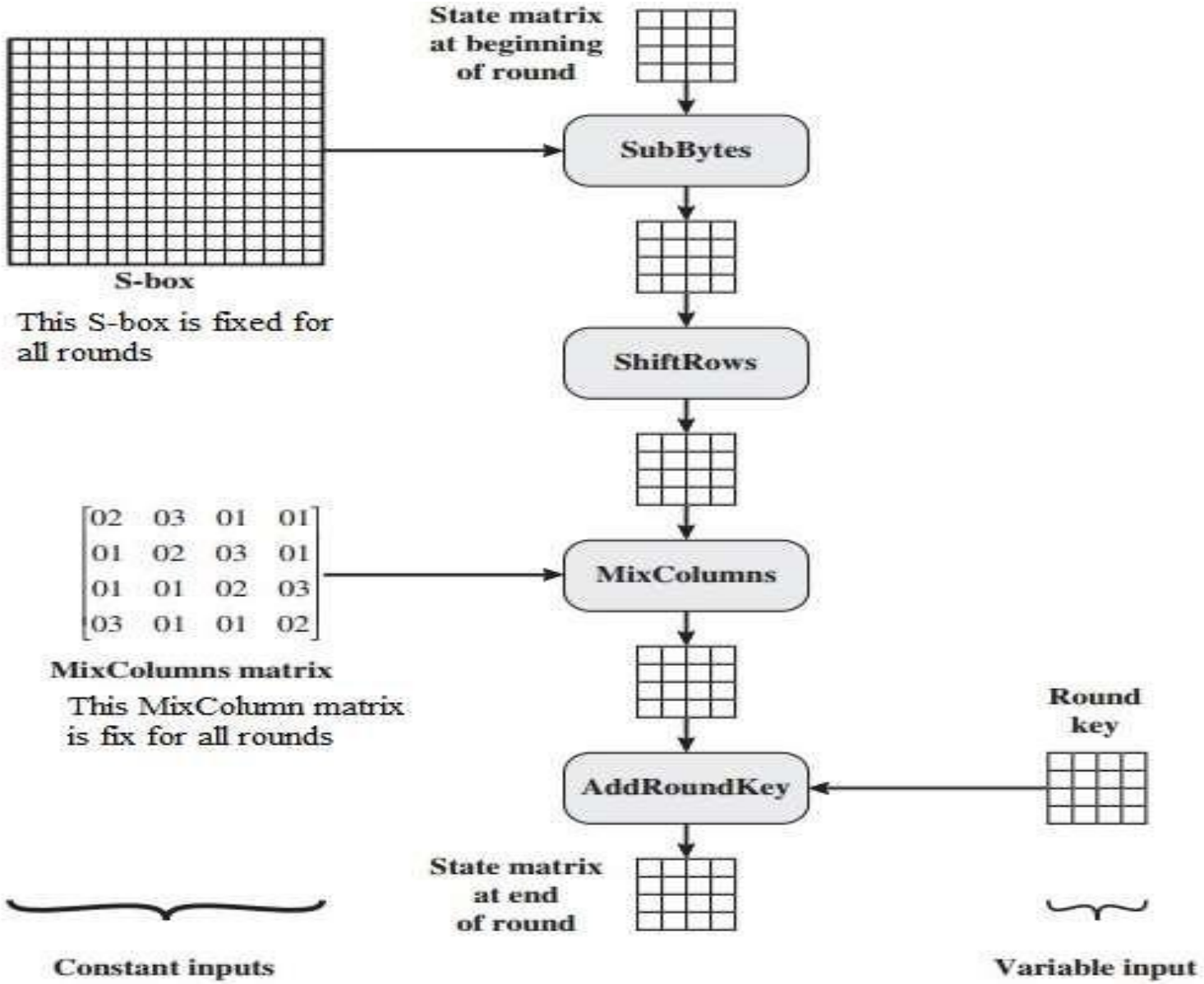
▪**Round 0:** Initial Transformation

In this round AddRoundKey operation is performed with plain text matrix of 128bits. Which means perform XOR operation between plain text matrix and Round 0 key i.e. w(0,3). The output of this stage is the input State Array matrix for Next Round.

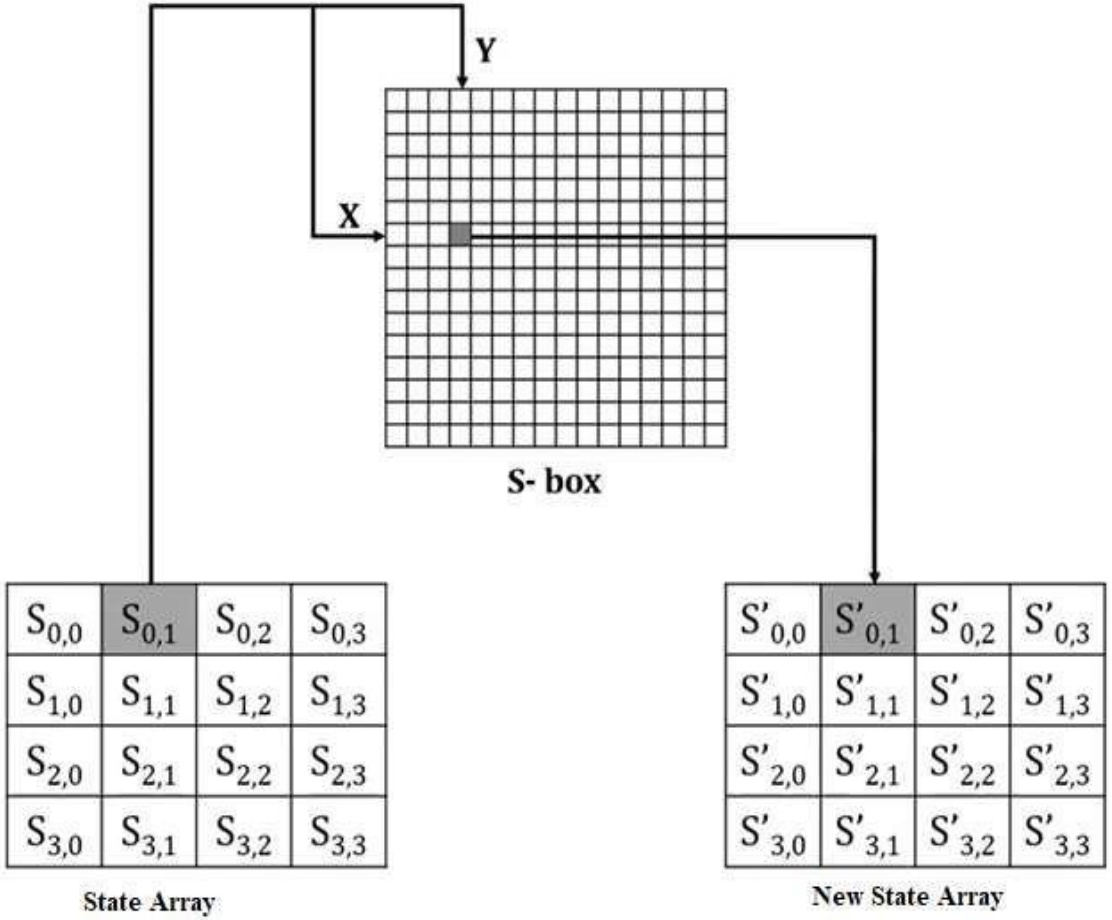
▪**Round 1:**

Four different stages are used in each round except last round which is not used MixColumn stage, one of permutation and three of substitution:

- **Substitute Bytes (SubBytes):** It Uses an fixed S-box matrix to perform a byte-by-byte substitution of blocks
- **ShiftRows:** A simple permutation
- **MixColumns:** A substitution that makes use of arithmetic over GF(28)
- **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key


State matrix at beginning of round
SubBytes
S-box
This S-box is fixed for all rounds
ShiftRows
02 03 01 01
01 02 03 01
01 01 02 03
03 01 01 02
MixColumns
MixColumns matrix
This MixColumn matrix is fix for all rounds
Round key
AddRoundKey
State matrix at end of round
Constant inputs
Variable input

- **Perform Sub Bytes Operation**



Suppose, $S_{0,1}$ = 63 it means map the value of S-box of row no 6 and col no 3. Note that rows and column always starts with 0.

Here, $S_{0,0}$ represent 0^{th} byte of 0^{th} word. Similarly, $S_{1,0}$ represent 1^{st} byte of 0^{th} word and so on.

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

(a) S-box

S-Box in AES which is used for encryption

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

(b) Inverse S-box

Inverse S-box in AES which is used in decryption

- **Example:** Suppose we have State Array matrix after initial transformation is:

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

Now perform substitute bytes using S-box. S-box is fixed for all the rounds.

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

1st Number E → Row
2nd Number A → Column

EA → 87

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

87			

Repeat this step for all the bytes in plain text matrix. The final result is:

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

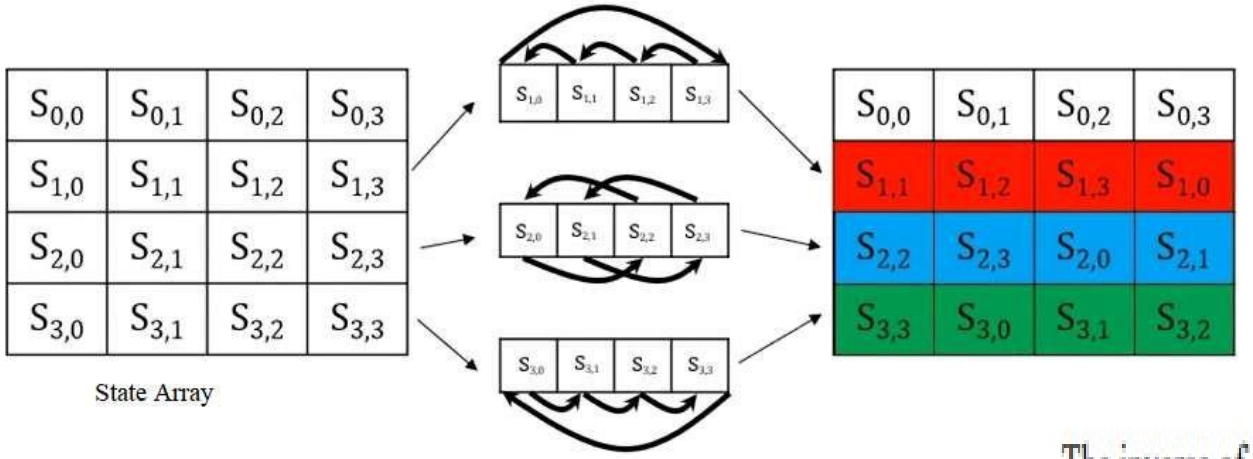
→

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

▪Perform ShiftRows Operation

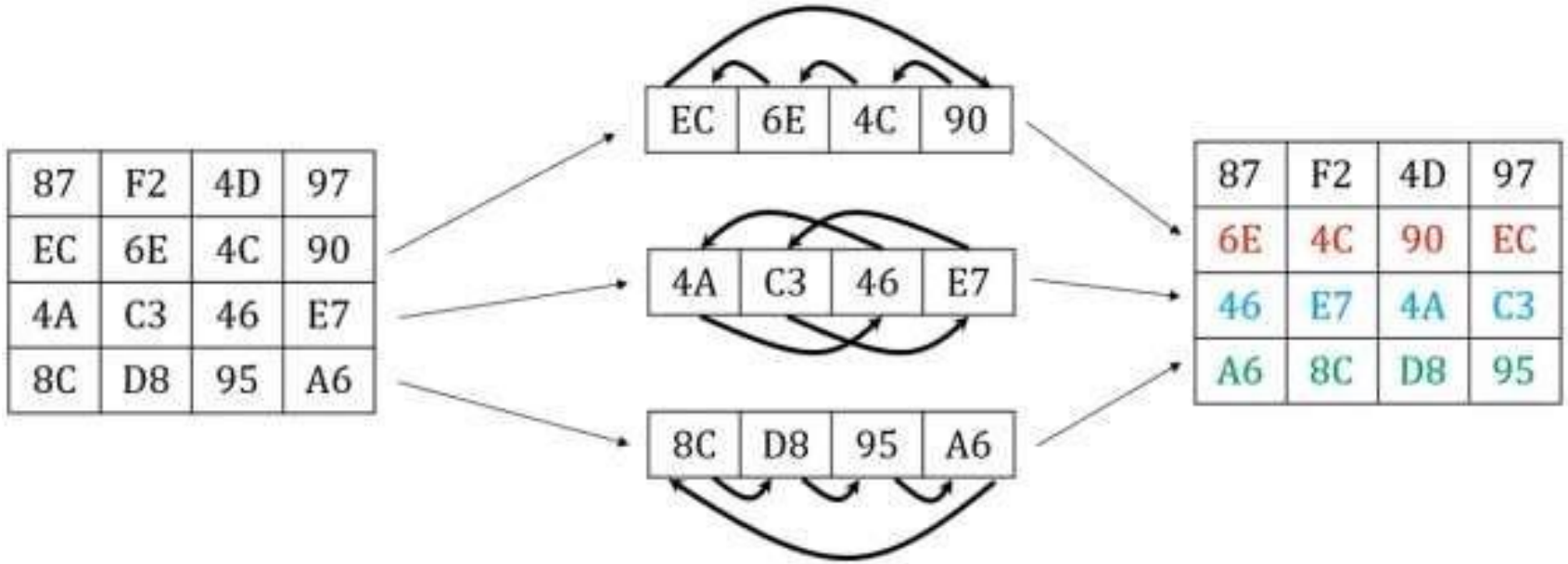
▪Left shift operation is performs in State Array. State Array is the output of Sub byte operation. Left shift operation is performs as:

- Row 1 - No Shifting
- Row 2 - 1 Byte Circular Left Shift
- Row 3 - 2 Byte Circular Left Shift
- Row 4 - 3 Byte Circular Left Shift



The inverse of Shift Row is the same cyclically shift but to the right.

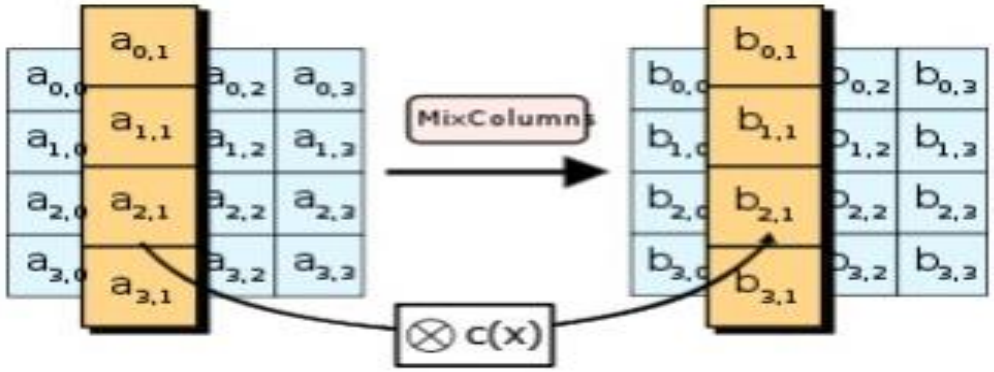
- **Example:** We have a State Array obtained in previous step



Perform MixColumn Operation

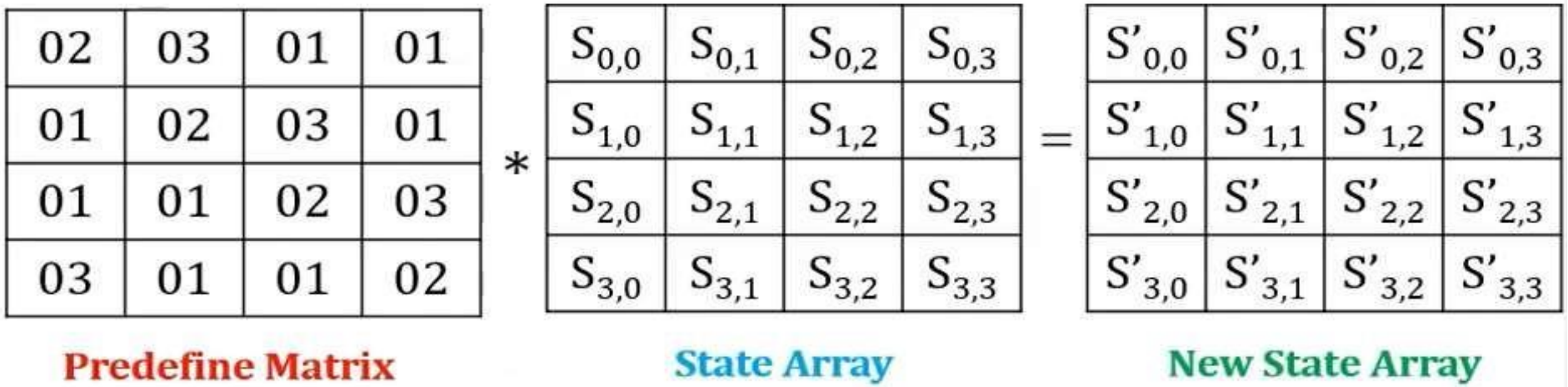
Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Each column of the state is multiplied with a fixed matrix. The multiplication is field multiplication in Galois field.



For this we have a predefined matrix. This is fixed matrix for all the rounds

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02



$$S'_{0,j} = (2 * S_{0,j}) \oplus (3 * S_{1,j}) \oplus S_{2,j} \oplus S_{3,j}$$

$$S'_{1,j} = S_{0,j} \oplus (2 * S_{1,j}) \oplus (3 * S_{2,j}) \oplus S_{3,j}$$

$$S'_{2,j} = S_{0,j} \oplus S_{1,j} \oplus (2 * S_{2,j}) \oplus (3 * S_{3,j})$$

$$S'_{3,j} = (3 * S_{0,j}) \oplus S_{1,j} \oplus S_{2,j} \oplus (2 * S_{3,j})$$

- Example for this matrix multiplication we have to use binary number system and polynomial theorem based on finite field theory or Galois Field theorem of GF(2^8).

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

*

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

=

?			

{02} * {87} ⊕ {03} * {6E} ⊕ {01} * {46} ⊕ {01} * {A6}

$X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + X^1 + 1$
0 0 0 0 0 0 1 0

02 = 0000 0010 = X
87 = 1000 0111 = $X^7 + X^2 + X + 1$

$X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + X^1 + 1$
1 0 0 0 0 1 1 1

Here, x^8 is not in the Galois Field of 2^8, maximum power of this field is x^7.

02 = 0000 0010 = X
87 = 1000 0111 = $X^7 + X^2 + X + 1$
02 * 87 = $X * (X^7 + X^2 + X + 1)$
= $X^8 + X^3 + X^2 + X$
= $X^4 + X^3 + X + 1 + X^3 + X^2 + X$
= $X^4 + X^2 + 1$
= 0001 0101

So breaking the values of x^8

- Similarly, find the product of {03}* {6E}, {01}* {46} and {01}* {A6},

$03 = 0000\ 0011 = X + 1$

$6E = 0110\ 1110 = X^6 + X^5 + X^3 + X^2 + X$

$03 * 6E = (X+1) * (X^6 + X^5 + X^3 + X^2 + X)$

$= X^7 + X^6 + X^4 + X^3 + X^2 + X^6 + X^5 + X^3 + X^2 + X$

$= X^7 + X^5 + X^4 + X$

$= 1011\ 0010$

$01 * 46 = 46 = 0100\ 0110$

$01 * A6 = A6 = 1010\ 0110$

Now, Perform XOR operation to multiplicative terms

For XOR, Odd numbers of 1's = 1 and Even numbers of 1's = 0

$02 * 87 = 0\ 0\ 0\ 1\ \ 0\ 1\ 0\ 1$

$03 * 6E = 1\ 0\ 1\ 1\ \ 0\ 0\ 1\ 0$

$01 * 46 = 0\ 1\ 0\ 0\ \ 0\ 1\ 1\ 0$

$01 * A6 = 1\ 0\ 1\ 0\ \ 0\ 1\ 1\ 0$

$= 0\ 1\ 0\ 0\ \ 0\ 1\ 1\ 1 = \{47\}$

$4 \quad\quad 7$

- ***First term*** = 47

- Likewise perform all the matrix multiplication. Final result is

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

*

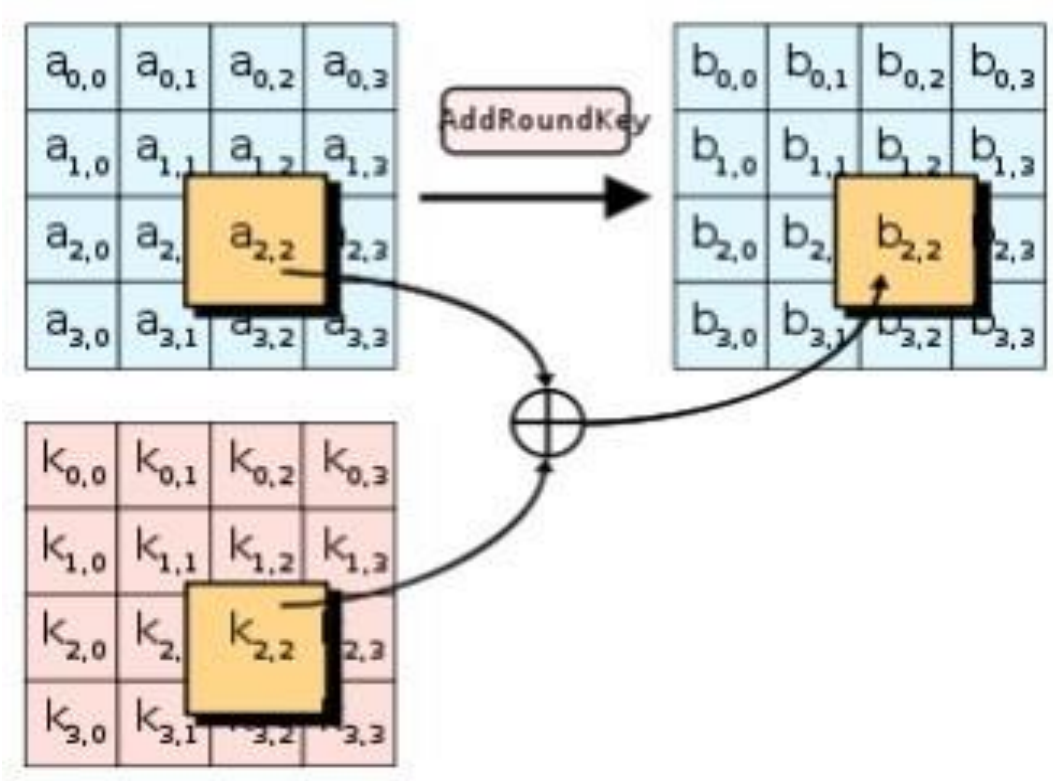
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

=

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

▪Perform AddRoundKey Operation

▪In this step we perform column wise XOR between State Array matrix and Round Key matrix. State Array matrix is the output of mix column in previous step.



- **Example:** Here, Round key is Round 1 Key i.e. w(4,7) which we derived in key expansion.

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

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB			
40			
F2			
1E			

47 $\oplus$ AC = ?
47 = 0100 0111
AC = 1010 1100
= 1110 1011
E B = {EB}

37 $\oplus$ 77 = ?
37 = 0011 0111
77 = 0111 0111
= 0100 0000
4 0 = {40}

94 $\oplus$ 66 = ?
94 = 1001 0100
66 = 0110 0110
= 1111 0010
F 2 = {F2}

ED $\oplus$ F3 = ?
ED = 1110 1101
F3 = 1111 0011
= 0001 1110
1 E = {1E}

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

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

- Finally first round is completed and we got the result of 1st round. This will be the input for next round.
- Repeat this up to 10 rounds and final State Array matrix is the Cipher text.
- Remember that final round i.e. 10th round for 128-bit key has only three stages where Mix Column stage is not performed.

PRIMITIVE ROOT

Integer 'a' is said to be a primitive root of prime number 'p' if $a^1 \ mod \ p, a^2 \ mod \ p, \ldots\ldots$ $\ldots\ldots, \ , a^{p-1} \ mod \ p$ are distinct and consists of integers from 1 to $p-1$ in some permutation.

Q. Is 2 a primitive root of 5?

Sol^n:

Here, $a = 2$ and $p = 5$

$2^1 \ mod \ 5 = 2$
$2^2 \ mod \ 5 = 4$
$2^3 \ mod \ 5 = 3$
$2^4 \ mod \ 5 = 1$

Here, all values are distinct and consists of integers 1 to 4.
$\therefore$ 2 is primitive root of 5.

Q. Find out whether 3 is primitive root of 7?

Sol^n:

Here, $a = 3$ and $p = 7$

$3^1 \ mod \ 7 = 3 \ mod \ 7 = 3$
$3^2 \ mod \ 7 = 9 \ mod \ 7 = 2$
$3^3 \ mod \ 7 = 27 \ mod \ 7 = 6$
$3^4 \ mod \ 7 = 81 \ mod \ 7 = 4$
$3^5 \ mod \ 7 = 243 \ mod \ 7 = 5$
$3^6 \ mod \ 7 = 729 \ mod \ 7 = 1$

Here, all values are distinct and consists of integers 1 to 6.

∴ 3 is primitive root of 7.

Q. Is 2 a primitive root of 7?

Sol^n:

Here, $a = 2$ and $p = 7$

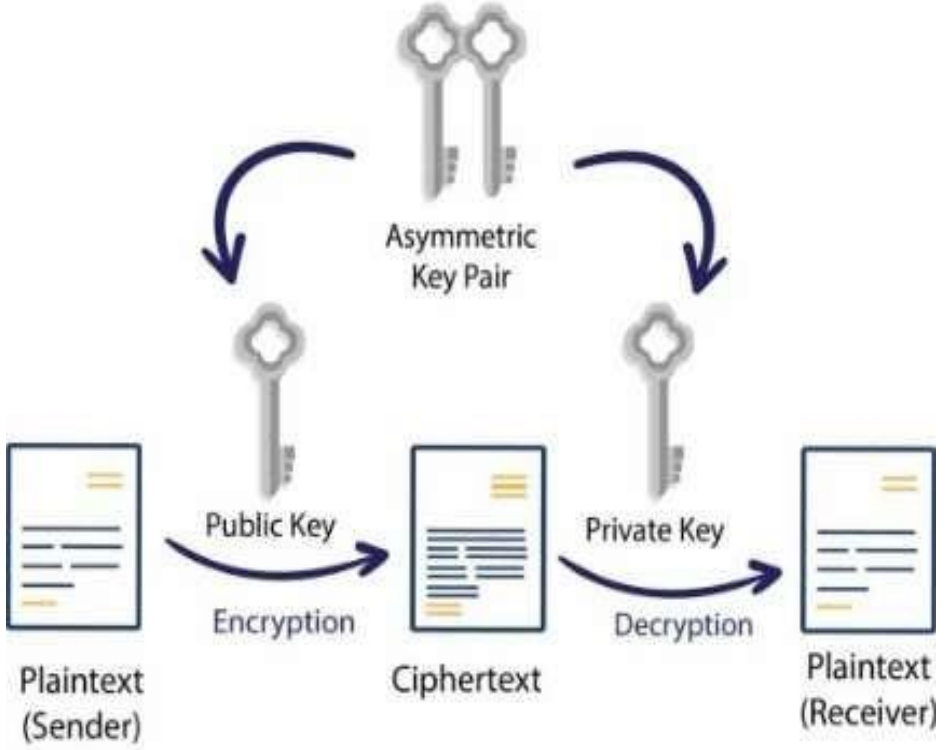
$2^1 \ mod \ 7 = 2$
$2^2 \ mod \ 7 = 4$
$2^3 \ mod \ 7 = 1$
$2^4 \ mod \ 7 = 2$

Repeat

∴ 2 is not primitive root of 7.

ASYMMETRIC ENCRYPTION

- Asymmetric encryption also called public key cryptography, works with a pair of keys. The beginning of asymmetric encryption involves the creation of a pair of keys, one of which is a public key, and the other which is a private key.
- The public key is accessible by anyone, while the private key must be kept a secret from everyone by the creator of the key. This is because encryption occurs with the public key, while decryption occurs with the private key. The recipient of the sensitive data will provide their public key to the sender, which will be used to encrypt the data. This ensures that only the recipient can decrypt the data, with their own private key.



DIFFIE-HELLMAN PROTOCOL

- The Diffie–Hellman (DH) Algorithm is not an encryption decryption algorithm. It is just a key-exchange protocol that enables two parties communicating over public channel to establish a mutual secret key without it being transmitted over the Internet. This is used in symmetric key cryptography.
- DH is not a symmetric algorithm – it is an asymmetric algorithm used to establish a shared secret for a symmetric key algorithm.
- The method of secret key generation is asymmetric cryptography or public key cryptography.

Algorithm:

Step 1: Select a prime number **q** which is public and shared by two parties **q** = any prime number

Step 2: Find the primitive root (α) of q and select one of them this is also public

Check for primitive root: $\alpha^n \bmod q$,

Where n = 1,2,3,….. (q-1) and the result should contains all the value less then q without repeated term.

Step 3: Generate private key (X) and public key (Y) for User A and User B

For User A:

Generate Private Key (X_A):

$X_A < q$

Where, X_A is any random number which is less then q ***Generate Public Key (Y_A):***

$Y_A = \alpha^{XA} \bmod q$

This public key (Y_A) is available for all the users.

For User B:

Generate Private Key (X_B):

$X_B < q$

Where, X_B is any random number which is less then q ***Generate Public Key (Y_B):***

$Y_B = \alpha^{XB} \bmod q$

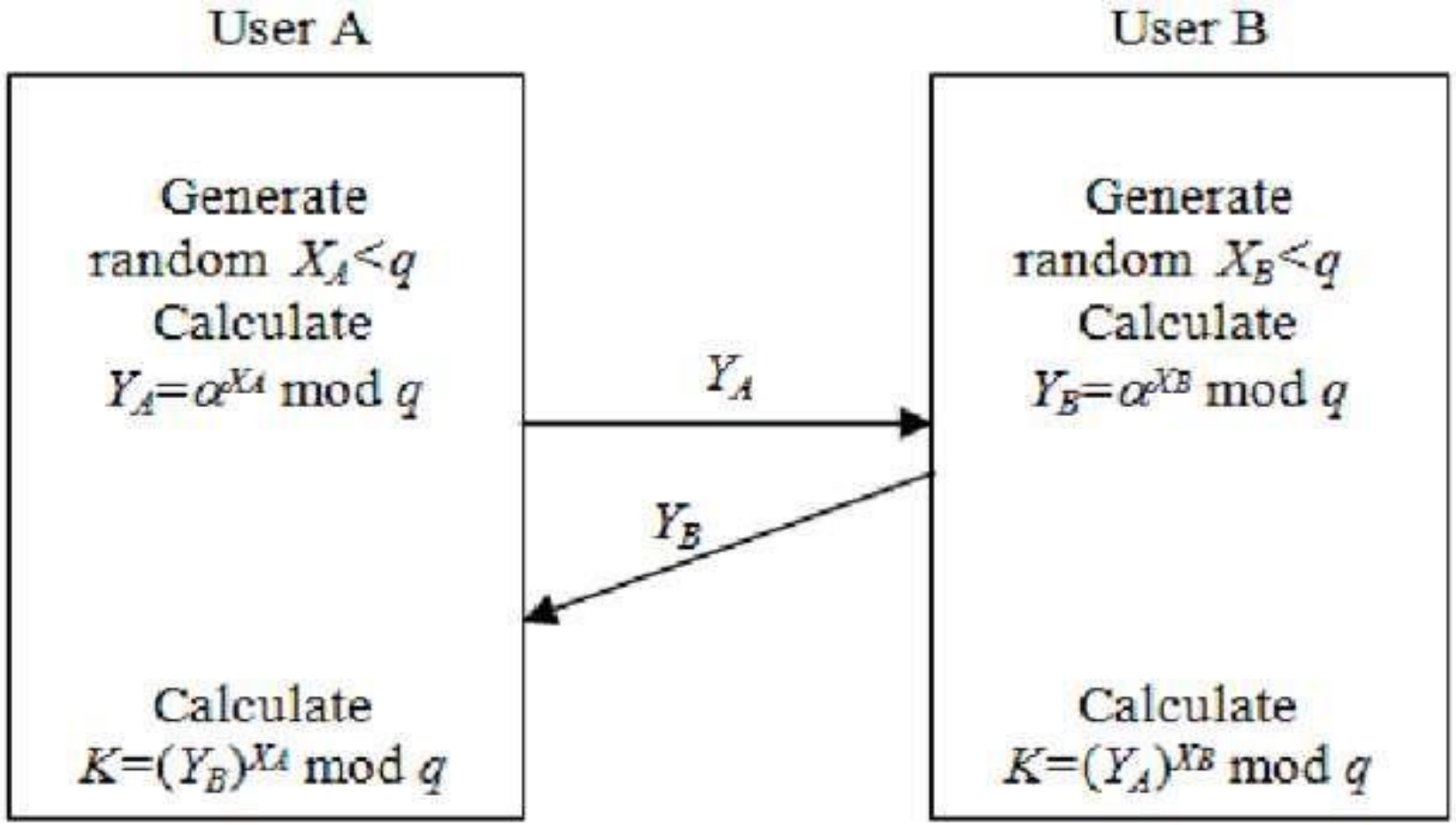
This public key (Y_B) is available for all the users.

Step 4: Generate a secret key for User A and User B

For User A: $K_A = (Y_B)^{XA} \bmod q$

For User B: $K_B = (Y_A)^{XB} \bmod q$

If $K_A = K_B$ then key exchange is successful otherwise fail.


User A
User B
Generate
random $X_A<q$
Calculate
$Y_A=\alpha^{X_A} \bmod q$
Generate
random $X_B<q$
Calculate
$Y_B=\alpha^{X_B} \bmod q$
Y_A
Y_B
Calculate
$K=(Y_B)^{X_A} \bmod q$
Calculate
$K=(Y_A)^{X_B} \bmod q$

Example: Generate a secret key for symmetric encryption **if prime number is 7.**

Step 1: Consider a prime number **q = 7**

Step 2: Find the primitive root (α) of q and select one of them ***Check for primitive root***

$\alpha^n \bmod q$,

Where n = 1,2,3,….. (q-1) and the result should contain all the value less then q without repeated term.

Let α = 2,

$2^1 \bmod 7 = 2$

$2^2 \bmod 7 = 4$

$2^3 \bmod 7 = 1$

$2^4 \bmod 7 = 2$ $2^5 \bmod 7 = 4$

$2^6 \bmod 7 = 1$

Here, all the number less than 7 is not covered, so 2 is not a primitive root of 7

Let α = 3,

$3^1 \bmod 7 = 3$

$3^2 \bmod 7 = 2$

$3^3 \bmod 7 = 6$

$3^4 \bmod 7 = 4$

$3^5 \bmod 7 = 5$

$3^6 \bmod 7 = 1$

Here, all the numbers less than 7 is covered, so 3 is a primitive root of 7

So, **α = 3**

Step 3: Generate private key (X) and Public Key (Y)

for User A and User B

For User A:

Generate Private key(X_A):

Choose a random number less than 7 as a private key

$X_A = 3$

Generate Public key(Y_A):

$Y_A = \alpha^{XA} \bmod q$

$= 3^3 \bmod 7$

$Y_A = 6$ (This is available publicly)

For User B:

Generate Private key(X_B):

Choose a random number less than 7 as a private key

$X_B = 5$

Generate Public key(Y_B):

$Y_B = \alpha^{XB} \bmod q$

$= 3^5 \bmod 7$

$Y_B = 5$ (This is available publicly)

Step 4: Generate Secret Key for both users

For User A:

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

$= 5^3 \bmod 7$

K_A = 6 *this is secret key generated by User A*

For User B:

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

$= 6^5 \bmod 7$

K_B = 6 *this is the secret key generated by User B*

Here, $K_A = K_B$ so key exchange is successful.

Sometimes the primitive root is directly given in Diffie Hellman Protocol

Steps

1. Generate two global public elements p and g, where p is prime number and $g < p$ is primitive root of p.
2. User A select random integer $X_A < p$ and computes $Y_A = g^{X_A} mod\ p$.
3. User B select random integer $X_B < p$ and computes $Y_B = g^{X_B} mod\ p$.
4. Each side keeps the X value as private and makes Y value available to each other.
5. User A computes key as $K = (Y_B)^{X_A}\ mod\ p$.
6. User B computes key as $K = (Y_A)^{X_B}\ mod\ p$.

Example

$p = 23, g = 5$

User A	User B
$X_A < p$, so user A chooses the secret integer $X_A = 6$.	$X_B < p$, so user B chooses the secret integer $X_B = 15$.
$Y_A = g^{X_A} mod\ p$	$Y_B = g^{X_B} mod\ p$
$= 5^6 mod\ 23$	$= 5^{15} mod\ 23$
$= 8$	$= 19$
User A sends the value of Y_A to user B.	User B sends the value of Y_B to user A.
$K = (Y_B)^{X_A}\ mod\ p$	$K = (Y_A)^{X_B}\ mod\ p$
$= 19^6\ mod\ 23$	$= 8^{15}\ mod\ 23$
$= 2$	$= 2$

Q. Consider a Deffie-Hellman scheme with a common prime $p = 11$ and a primitive root $g = 2$.
i) Show that 2 is a primitive root of 11.
ii) If user A has public key $Y_A = 9$, what is A's private key X_A?
iii) If user B has public key $Y_B = 3$, what is shared key K, shared with A.

Solution:

i) Here, $p = 11$ & $g = 2$

$2^1 \ mod \ 11 = 2 \ mod \ 11 = 2$
$2^2 \ mod \ 11 = 4 \ mod \ 11 = 4$
$2^3 \ mod \ 11 = 8 \ mod \ 11 = 8$
$2^4 \ mod \ 11 = 16 \ mod \ 11 = 5$
$2^5 \ mod \ 11 = 32 \ mod \ 11 = 10$
$2^6 \ mod \ 11 = 64 \ mod \ 11 = 9$
$2^7 \ mod \ 11 = 128 \ mod \ 11 = 7$
$2^8 \ mod \ 11 = 256 \ mod \ 11 = 3$
$2^9 \ mod \ 11 = 512 \ mod \ 11 = 6$
$2^{10} \ mod \ 11 = 1024 \ mod \ 11 = 1$

Here all values are distinct and consists of integers 1 to 10 (i.e. 1 to $p - 1$). Hence, 2 is a primitive root of 11.

ii) User A's public key $Y_A = 9$

A's private key $X_A = ?$

We have,

$$Y_A = g^{X_A} mod\ p$$

$$9 = 2^{X_A}\ mod\ 11$$

From this equation,

$X_A = 6$, because $2^6\ mod\ 11 = 9$.

$\therefore$ ***A's private key*** $\mathbf{X_A = 6.}$

iii) B's public key $Y_B = 3$

Now,

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

$$= 3^6\ mod\ 11$$

$$= 729\ mod\ 11$$

$$= 3$$

$\therefore$ ***Shared key*** $\mathbf{K = 3.}$

RSA ALGORITHM

- The RSA algorithm (Rivest-Shamir-Adleman) is the basis of a cryptosystem that are used for specific security services or purposes which enables public key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network such as the internet.

- It is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private Key is kept private.

- An example of asymmetric cryptography:

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. The client receives this data and decrypts it.

- Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

- RSA algorithm is public key cryptography i.e. it works on two different keys i.e. public key and private key.
- The public key can be known to everyone and is used for encrypting message. Message encrypted with the public key can only be decrypted using the private key.

Algorithm

RSA key generation:

1. Choose two distinct large prime numbers p *and* q.
2. Compute $n = pq$, n is used as modulus for both public and private keys.
3. Compute the totient: $\phi(n) = (p-1)(q-1)$.
4. Choose an integer e such that $1 < e < \phi(n)$ and e *and* $\phi(n)$ are co-prime.
5. Compute d to satisfy $ed \equiv 1 (mod\ \phi(n))$. (ed mod φ(n) =1) ; i.e. de = 1 + kφ(n) for some integer k)
6. Public key is $\{e, n\}$.
7. Private key is $\{d, n\}$.

Encryption:

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

Decryption:

$$m = c^d \bmod n$$

Example

$p = 5 \,\&\, q = 19$

$n = pq = 5 * 19 = 95$

$\phi(n) = (5-1)(19-1) = 4 * 18 = 72$

Choose e, such that $1 < e < 72$ and co-prime to 72.

$\therefore e = 5$

Calculate d by using;

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

$5 * d \equiv 1(mod\ 72)$

$5 * 29 \equiv 1(mod\ 72)$

$\therefore d = 29$

So, the public key is $\{e, n\} = \{5, 95\}$ and the private key is $\{d, n\} = \{29, 95\}$.

Consider $m = 19$

Encryption:

$c = m^e\ mod\ n$

$= 19^5\ mod\ 95$

$= 19$

Decryption:

$m = c^d\ mod\ n$

$= 19^{29}\ mod\ 95$

$= 19$

Q. In a RSA system, a user has chosen the primes 53 and 59 to create a key pair. Now show that the generation of public key pair (e, n) and private key pair (d, n). Show encryption and decryption process for the message "HI".

Solution:

Given,

$p = 53, q = 59$

$n = pq = 53 * 59 = 3127$

$\phi(n) = (53 - 1)(59 - 1) = 3016$

Choose e, such that $1 < e < 3016$ and co-prime to 3016.

$\therefore e = 3$

Calculate d by using;

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

$3 * d \equiv 1(mod\ 3016)$

$3 * 2011 \equiv 1(mod\ 3016)$

$\therefore d = 2011$

So, the public key is $\{e, n\} = \{3, 3127\}$ and the private key is $\{d, n\} =$ {2011,3127}

Let us assume "*HI*" = 89

Encryption:

$c = m^e\ mod\ n$

$= 89^3\ mod\ 3127$

$= 1394$

Decryption:

$m = c^d\ mod\ n$

$= 1394^{2011}\ mod\ 3127$

$= 89$

Example: Consider, p = 61 and q = 53 now, compute n = pq = 61×53 = 3233

Compute the totient $\varphi(n) = (p - 1)(q - 1) = (61-1)(53-1) = 3120$

Choose e > 1 relatively prime to 3120; e = 17

Compute d such that $ed \equiv 1 \bmod \varphi(n)$ e.g., by computing the modular multiplicative inverse of e modulo φ(n): d = 2753 since 17×2753 = 46801 = 1 + 15×3120.

The public key is (n = 3233, e = 17).

For a padded message m the encryption function is:

$c = m^e$ **mod** n = m^{17} **mod** 3233.

The private key is (n = 3233, d = 2753). The decryption function is:

$m = c^d$ **mod** n = c^{2753} **mod** 3233.

- For example, to encrypt m = 123, we calculate
- $c = 123^{17}$ **mod** 3233 = 855 to decrypt c = 855, we calculate m = 855^{2753} **mod** 3233 = 123