Chapter 2

Cryptographic Algorithms

Table of Contents

Classical Cryptography:

Classical cryptography refers to the study and practice of securing information through traditional methods of encryption and decryption, which were primarily developed and used before the advent of digital computers.

It encompasses a variety of techniques and systems designed to protect sensitive information from unauthorized access.

Classical Encryption Techniques:

There are two basic building blocks of all encryption techniques:

1. substitution
2. transposition

1. Substitution Technique:

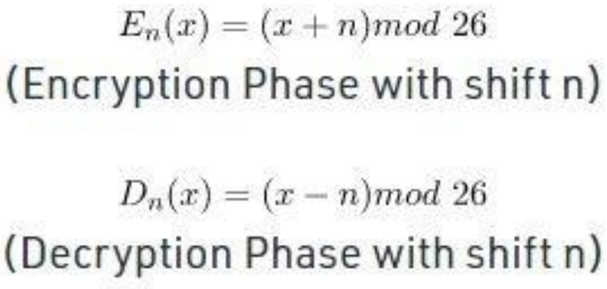
It is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

a. Caesar Cipher:

The Caesar cipher, named after Julius Caesar, is one of the simplest and most well-known encryption techniques.

It is a type of substitution cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet.

How Caesar Cipher Works



$$E_n(x) = (x + n) mod\ 26$$

(Encryption Phase with shift n)

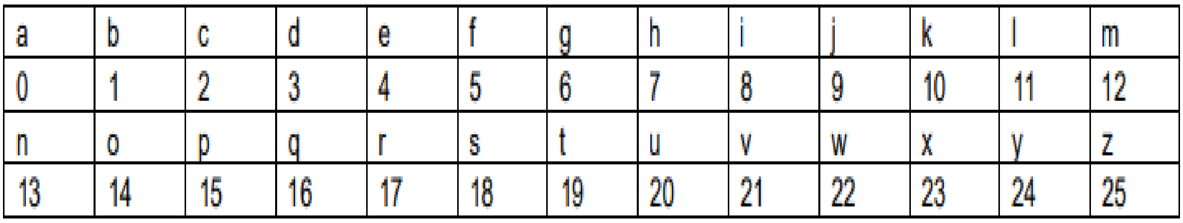
$$D_n(x) = (x - n) mod\ 26$$

(Decryption Phase with shift n)

Example:

Plaintext: meet me after the party

Ciphertext: PHHW PH DIWHU WKH SDUWB



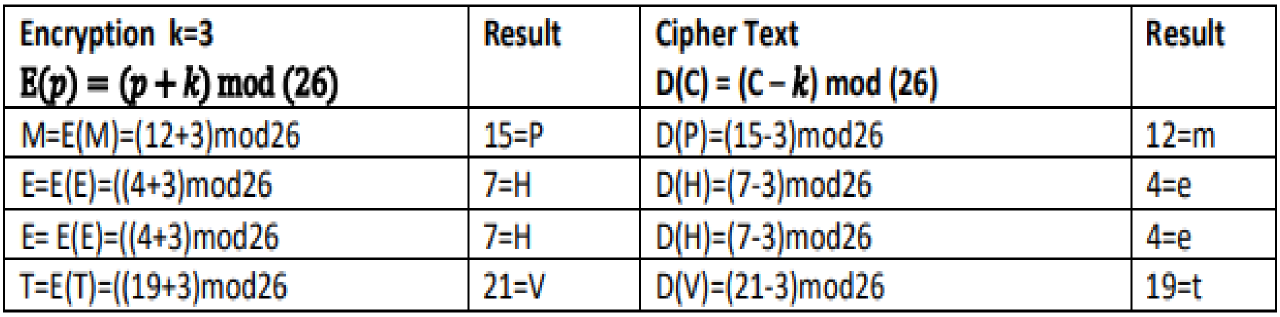
a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Here, the key is 3. If different key is used, different substitution will be obtained.

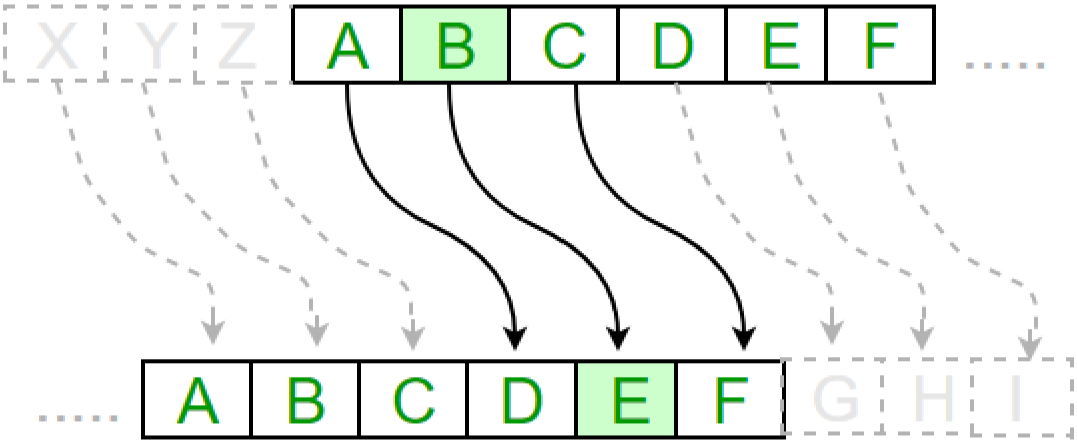
Mathematically, starting from a=0, b=1 and so on, Caesar cipher can be written as:

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

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



Encryption k=3 $E(p) = (p + k) \bmod (26)$	Result	Cipher Text $D(C) = (C - k) \bmod (26)$	Result
M=E(M)=(12+3)mod26	15=P	D(P)=(15-3)mod26	12=m
E=E(E)=((4+3)mod26	7=H	D(H)=(7-3)mod26	4=e
E= E(E)=((4+3)mod26	7=H	D(H)=(7-3)mod26	4=e
T=E(T)=((19+3)mod26	21=V	D(V)=(21-3)mod26	19=t



Examples :

Text : ATTACKATONCE

Shift: 4

Cipher: EXXEGOEXSRGI

Advantages:

- Easy to implement
- Requires only small pre-shared information
- Can be modified easily to make more secure, like using multiple shifts

Disadvantages:

- It is not secure against modern decryption methods
- Doesn’t provide confidentiality, integrity, and authenticity in a message
- Not useful for long messages as it can be cracked more easily

b. Vigenere Cipher:

Vigenere Cipher is a method of encrypting alphabetic text.

It uses a simple form of polyalphabetic substitution.

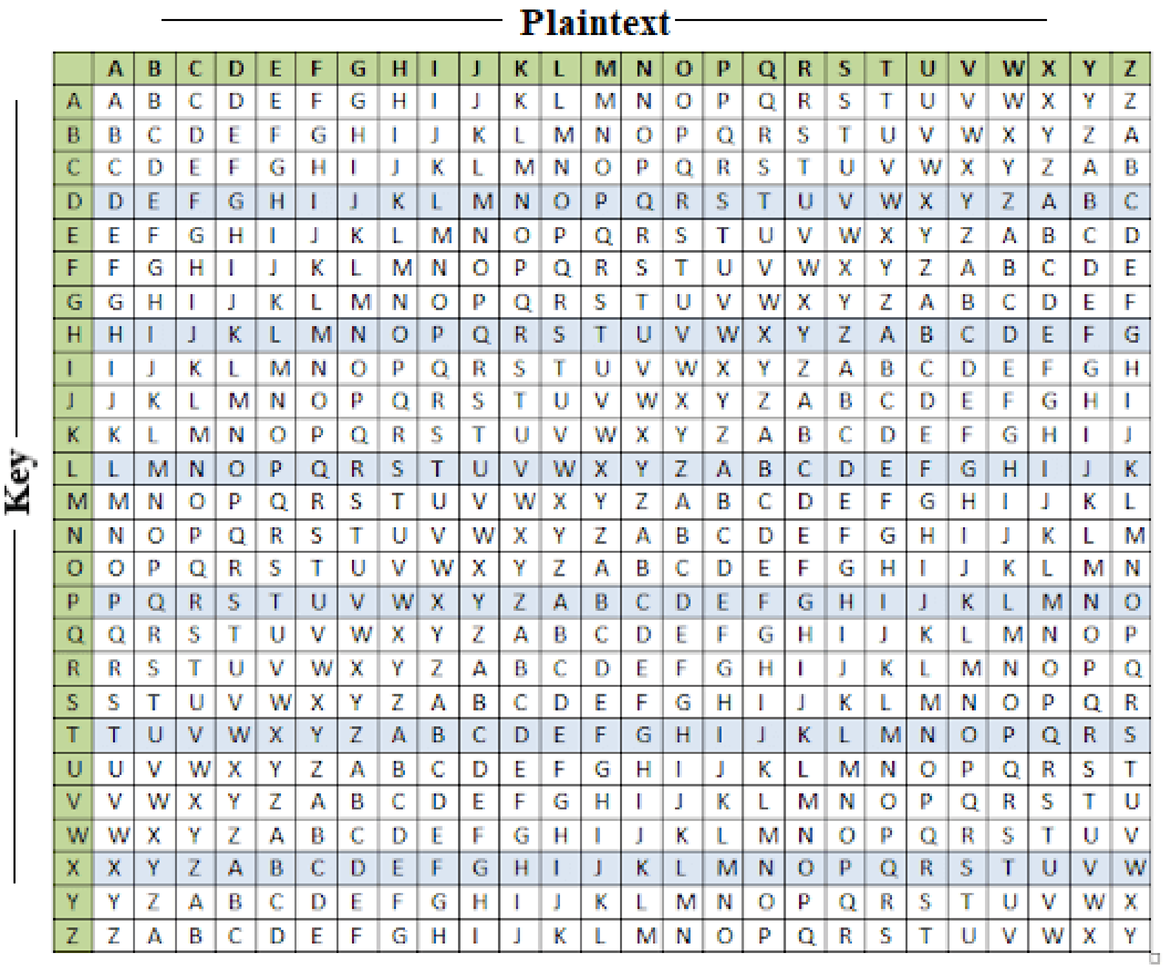
A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets.

The encryption of the original text is done using the *Vigenère square or Vigenère table*.

Two methods perform the vigenere cipher:

Method 1

When the vigenere table is given, the encryption and decryption are done using the vigenere table (26 * 26 matrix) in this method.



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: The plaintext is "JAVATPOINT", and the key is "BEST".

To generate a new key, the given key is repeated in a circular manner, as long as the length of the plain text does not equal to the new key.

J	A	V	A	T	P	O	I	N	T
B	E	S	T	B	E	S	T	B	E

Encryption

The first letter of the plaintext is combined with the first letter of the key. The column of plain text "J" and row of key "B" intersects the alphabet of "K" in the vigenere table, so the first letter of ciphertext is "K".

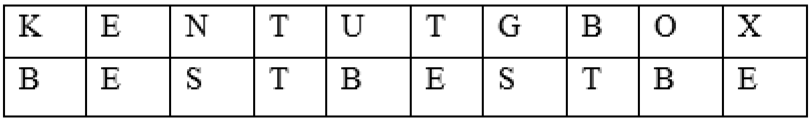
Similarly, the second letter of the plaintext is combined with the second letter of the key. The column of plain text "A" and row of key "E" intersects the alphabet of "E" in the vigenere table, so the second letter of ciphertext is "E".

This process continues continuously until the plaintext is finished.

Ciphertext = KENTUTGBOX

Decryption

Decryption is done by the row of keys in the vigenere table. First, select the row of the key letter, find the ciphertext letter's position in that row, and then select the column label of the corresponding ciphertext as the plaintext.



K	E	N	T	U	T	G	B	O	X
B	E	S	T	B	E	S	T	B	E

For example, in the row of the key is "B" and the ciphertext is "K" and this ciphertext letter appears in the column "J", that means the first plaintext letter is "J".

Next, in the row of the key is "E" and the ciphertext is "E" and this ciphertext letter appears in the column "A", that means the second plaintext letter is "A".

This process continues continuously until the ciphertext is finished.

Plaintext = JAVATPOINT

Method 2

When the vigenere table is not given, the encryption and decryption are done by Vigenar algebraically formula in this method (convert the letters (A-Z) into the numbers (0-25)).

Formula of encryption is,

$E_i = (P_i + K_i) \bmod 26$

Formula of decryption is,

$D_i = (E_i - K_i) \bmod 26$

If any case (D_i) value becomes negative (-ve), in this case, we will add 26 in the negative value.

Where,

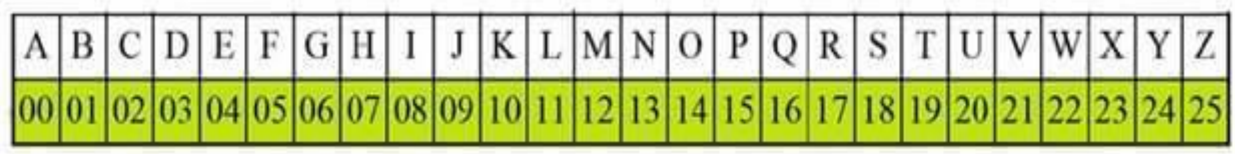
E denotes the encryption.

D denotes the decryption.

P denotes the plaintext.

K denotes the key.

Note: "i" denotes the offset of the ith number of the letters, as shown in the table below.



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

Example: The plaintext is "JAVATPOINT", and the key is "BEST".

Encryption: $E_i = (P_i + K_i) \bmod 26$

Plaintext	J	A	V	A	T	P	O	I	N	T
Plaintext value (P)	09	00	21	00	19	15	14	08	13	19
Key	B	E	S	T	B	E	S	T	B	E
Key value (K)	01	04	18	19	01	04	18	19	01	04
Ciphertext value (E)	10	04	13	19	20	19	06	01	14	23
Ciphertext	K	E	N	T	U	T	G	B	O	X

Decryption: $D_i = (E_i - K_i) \bmod 26$

If any case (Di) value becomes negative (-ve), in this case, we will add 26 in the negative value. Like, the third letter of the ciphertext;

N = 13 and S = 18

$D_i = (E_i - K_i) \bmod 26$

$D_i = (13 - 18) \bmod 26$

$D_i = -5 \bmod 26$

$D_i = (-5 + 26) \bmod 26$

$D_i = 21$

Ciphertext	K	E	N	T	U	T	G	B	O	X
Ciphertext value (E)	10	04	13	19	20	19	06	01	14	23
Key	B	E	S	T	B	E	S	T	B	E
Key value (K)	01	04	18	19	01	04	18	19	01	04
Plaintext value (P)	09	00	21	00	19	15	14	08	13	19
Plaintext	J	A	V	A	T	P	O	I	N	T

c. Playfair Cipher:

Playfair cipher is proposed by Charles Whetstone in 1889.

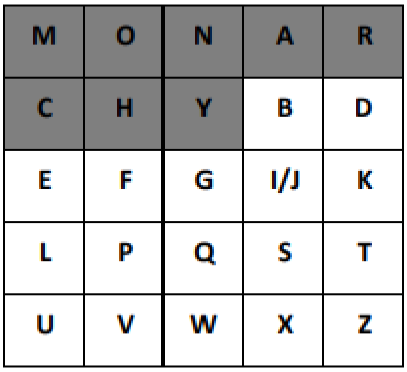
But it was named for one of his friends Lord Lyon Playfair because he popularized its uses.

It is the most popular symmetric encryption technique that falls under the substitution cipher.

It is an encoding procedure that enciphers more than one letter at a time.

In this technique multiple (2) letters are encrypted at a time.

This technique uses a 5 X 5 matrix which is also called key matrix:



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

Playfair Cipher Encryption Rules:

1. First, split the plaintext into digraphs (pair of two letters). If the plaintext has the odd number of letters, append the letter Z at the end of the plaintext. It makes the plaintext of even For example, the plaintext MANGO has five letters. So, it is not possible to make a digraph. Since, we will append a letter Z at the end of the plaintext, i.e. MANGOZ.
2. After that, break the plaintext into digraphs (pair of two letters). If any letter appears twice (side by side), put X at the place of the second occurrence. Suppose, the plaintext is COMMUNICATE then its digraph becomes CO MX MU NI CA TE. Similarly, the digraph for the plaintext JAZZ will be JA ZX ZX, and for plaintext GREET, the digraph will be GR EX ET.
3. To determine the cipher (encryption) text, first, build a 5*5 key-matrix or key-table and filled it with the letters of alphabets, as directed below:
4. If both letters fall in the same row of the key matrix, replace each with the letter to its right (wrapping back to start from end), eg. “AR" encrypts as "RM".

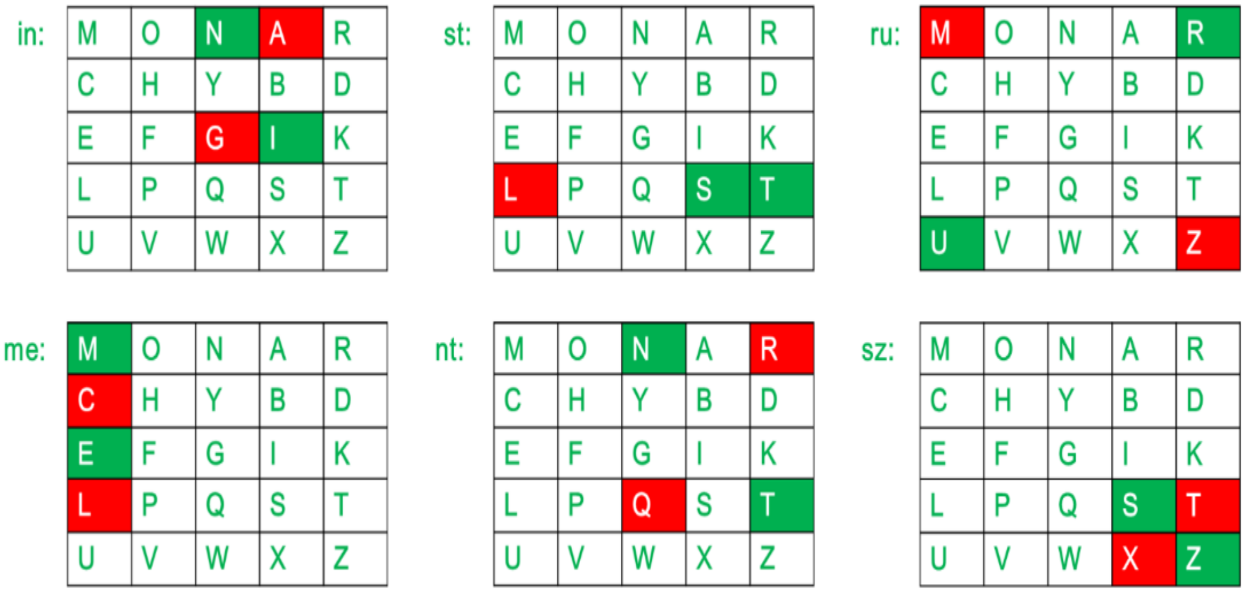
5. If both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom), eg. “MU" encrypts to "CM".
6. Otherwise each letter is replaced by the one in its row in the column of the other letter of the pair, eg. “HS" encrypts to "BP", and “EA" to "IM" or "JM" (as desired)

Example : PlainText: "instruments" keyword: monarchy

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

cipher text : ga tl mz cl rq tx

For both encryption and decryption, the same key is to be used.



in:

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

st:

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

ru:

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

me:

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

nt:

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

sz:

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

Decryption Technique

Decrypting the Playfair cipher is as simple as doing the same process in reverse. The receiver has the same key and can create the same key table, and then decrypt any messages made using that key.

1. **For example:**

   CipherText: "gatlmzclrqtx"

   After Split: 'ga' 'tl' 'mz' 'cl' 'rq' 'tx'

1. **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).
   - **For example:**

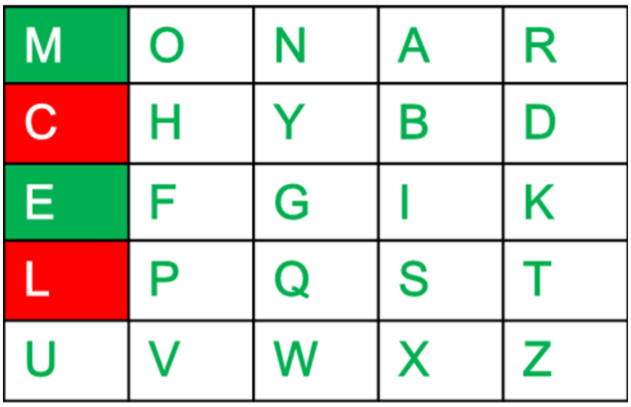
     Diagraph: "cl"

     Decrypted Text: me

     Decryption:

     c -> m

     l -> e



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

- **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).

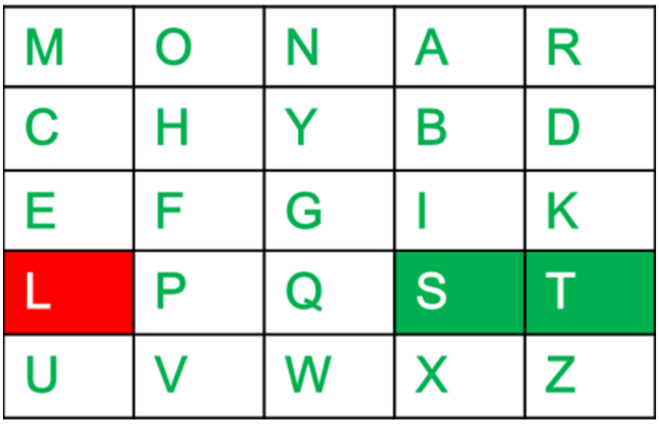
  Diagraph: "tl"

  Decrypted Text: st

  Decryption:

  t -> s

  l -> t



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

- **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.
- **For example:**

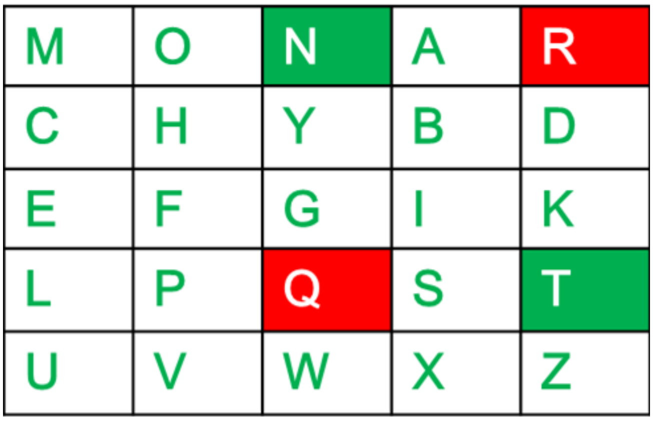
  Diagraph: "rq"

  Decrypted Text: nt

  Decryption:

  r -> n

  q -> t



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

-

For example:

Plain Text: "gatlmzclrqtx"

Decrypted Text: instrumentsz

Decryption:

(red)-> (green)

ga -> in

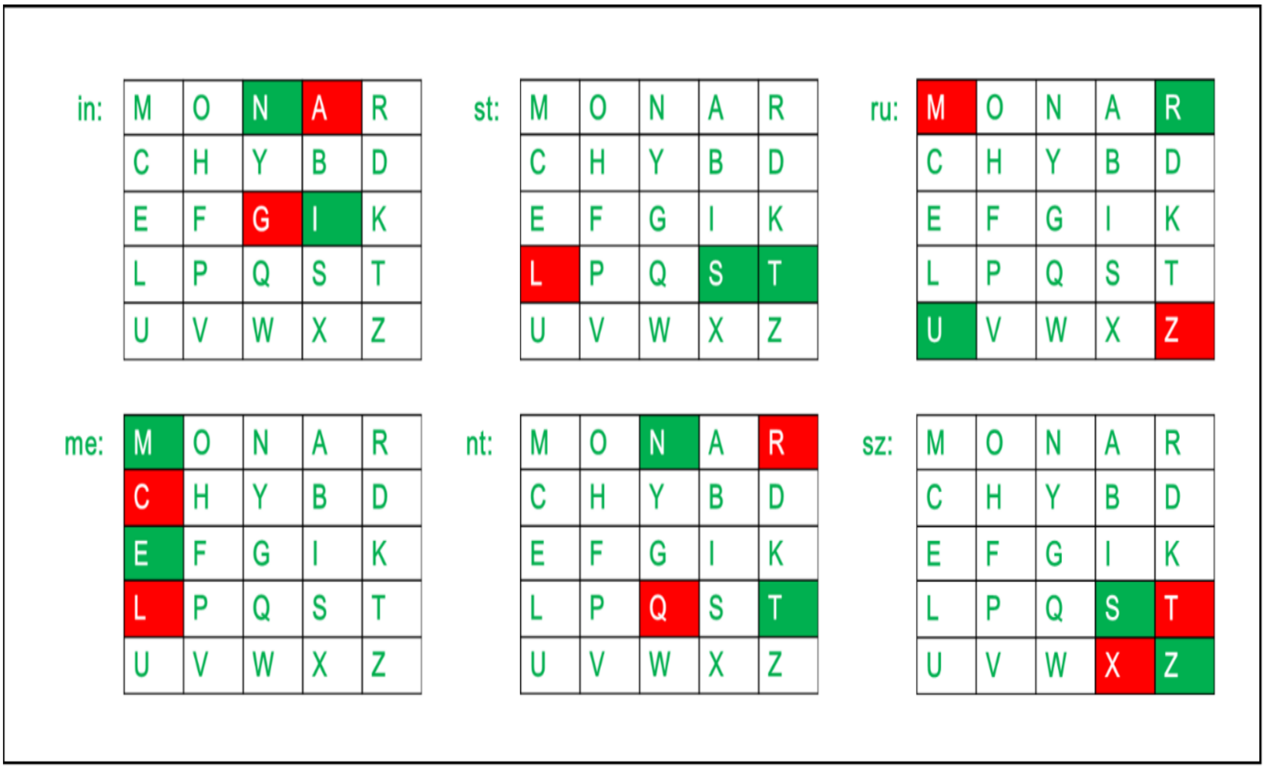
tl -> st

mz -> ru

cl -> me

rq -> nt

tx -> sz



in:

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

st:

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

ru:

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

me:

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

nt:

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

sz:

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

2. Transposition ciphers:

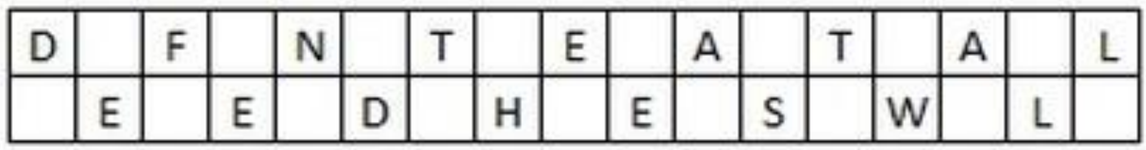
Transposition ciphers are a type of classical encryption technique where the positions of the characters in the plaintext are shifted according to a regular system to form the ciphertext.

Unlike substitution ciphers, which replace characters with other characters, transposition ciphers retain the original characters but change their order.

a. Rail Fence cipher:

The Rail Fence cipher works by writing your message on alternate lines across the page, and then reading off each line in turn.

For example, the plaintext "defend the east wall" is written as shown below, with all spaces removed.



D		F		N		T		E		A		T		A		L
	E		E		D		H		E		S		W		L	

The simplest Rail Fence Cipher, where each letter is written in a zigzag pattern across the page.

The ciphertext is then read off by writing the top row first, followed by the bottom row, to get "DFNTEATALEEDHESWL".

Encryption

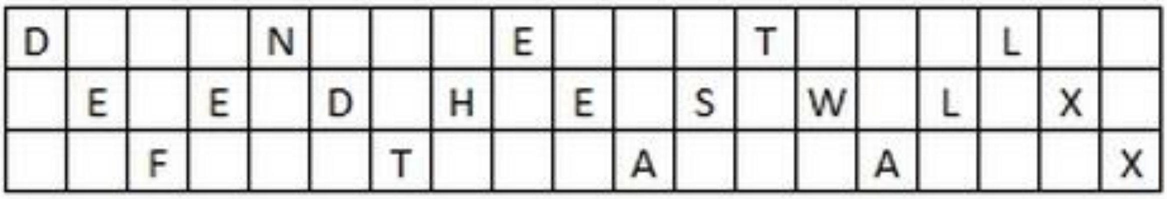
To encrypt a message using the Rail Fence Cipher, you have to write your message in zigzag lines across the page, and then read off each row.

Firstly, you need to have a key, which for this cipher is the number of rows you are going to have.

You then start writing the letters of the plaintext diagonally down to the right until you reach the number of rows specified by the key.

You then bounce back up diagonally until you hit the first row again. This continues until the end of the plaintext.

For the plaintext we used above, "defend the east wall", with a key of 3, we get the encryption process shown below.



D				N				E				T				L		
	E		E		D		H		E		S		W		L		X	
		F				T				A				A				X

The Rail Fence Cipher with a key of 3. Notice the nulls added at the end of the message to make it the right length.

Note that at the end of the message we have inserted two "X"s. These are called nulls, and act as placeholders.

The ciphertext is read off row by row to get "DNETLEEDHESWLXFTAAX".

Decryption

The decryption process for the Rail Fence Cipher involves reconstructing the diagonal grid used to encrypt the message.

We start writing the message, but leaving a dash in place of the spaces yet to be occupied.

Gradually, you can replace all the dashes with the corresponding letters, and read off the plaintext from the table.

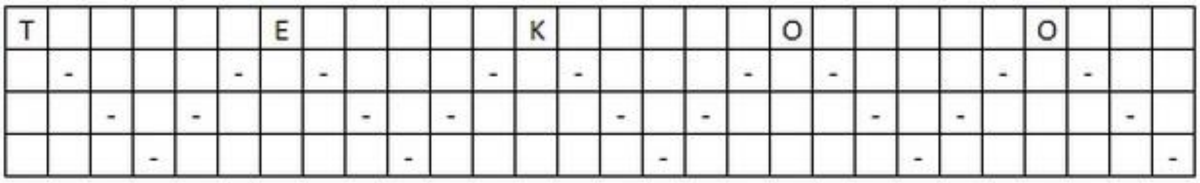
We then place the first letter in the top left square, and dashes diagonally downwards where the letters will be.

When we get back to the top row, we place the next letter in the ciphertext.

Continue like this across the row, and start the next row when you reach the end.

For example, if you receive the ciphertext "TEKOOHRACIRMNREATANFTETYTGHH", encrypted with a key of 4, you start by placing the "T" in the first square.

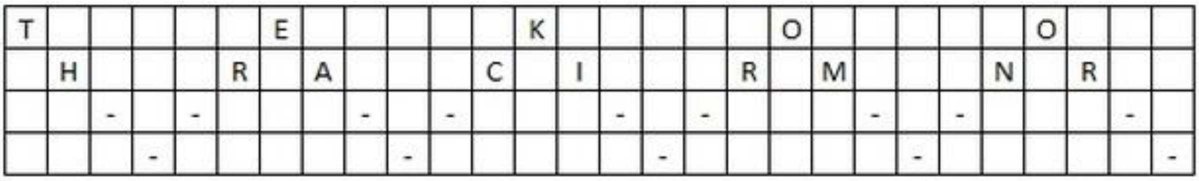
You then dash the diagonal down spaces until you get back to the top row, and place the "E" here. Continuing to fill the top row you get the pattern below.



T						E						K						O						O			
	-				-		-				-		-				-		-				-		-		
		-		-				-		-				-		-				-		-				-	
			-						-						-						-						-

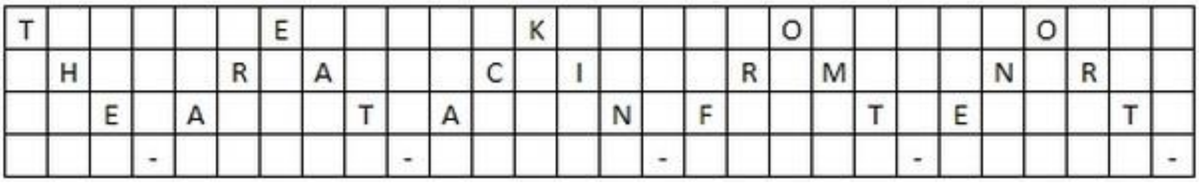
The first row of the decryption process for the Rail Fence Cipher. We have a table with 4 rows because the key is 4, and 28 columns as the ciphertext has length 28.

Continuing this row-by-row, we get the successive stages shown below.



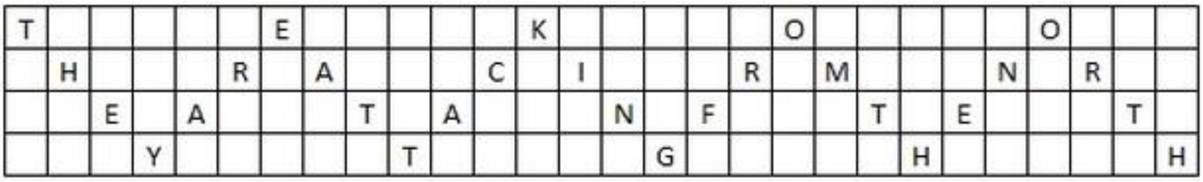
T						E						K						O						O			
	H				R		A				C		I				R		M				N		R		
		-		-				-		-				-		-				-		-				-	
			-						-						-						-						-

The second stage in the decryption process.



T						E						K						O						O			
	H				R		A				C		I				R		M				N		R		
		E		A				T		A				N		F				T		E				T	
			-						-						-						-						-

The third stage in the decryption process.



T						E						K						O						O			
	H				R		A				C		I				R		M				N		R		
		E		A				T		A				N		F				T		E				T	
			Y						T						G						H						H

The fourth and final stage in the decryption process.

From this we can now read the plaintext off following the diagonals to get "they are attacking from the north".

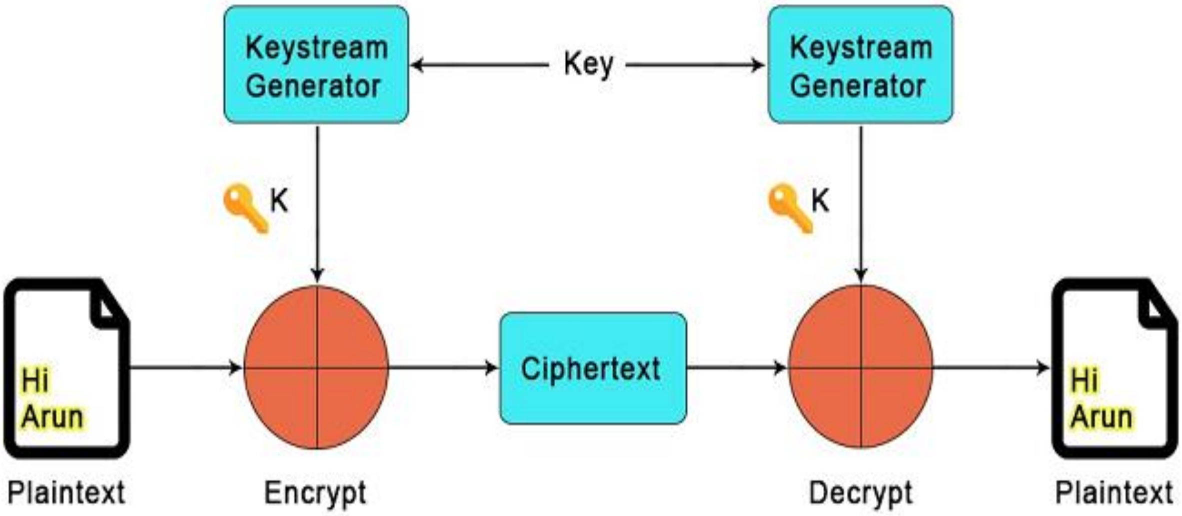
Modem Ciphers:

Modern cryptography uses more advanced and secure techniques compared to classical encryption methods.

Stream Ciphers

Stream ciphers encrypt plaintext one bit or byte at a time, producing a corresponding bit or byte of ciphertext.

They generate a keystream, a sequence of bits used to encrypt the plaintext by combining it with the keystream using XOR.



Encryption

For Encryption,

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

Example:

Plain Text : 10011001

Keystream : 11000011

Cipher Text : 01011010

Decryption

For Decryption,

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

Example:

Cipher Text : 01011010

Keystream : 11000011

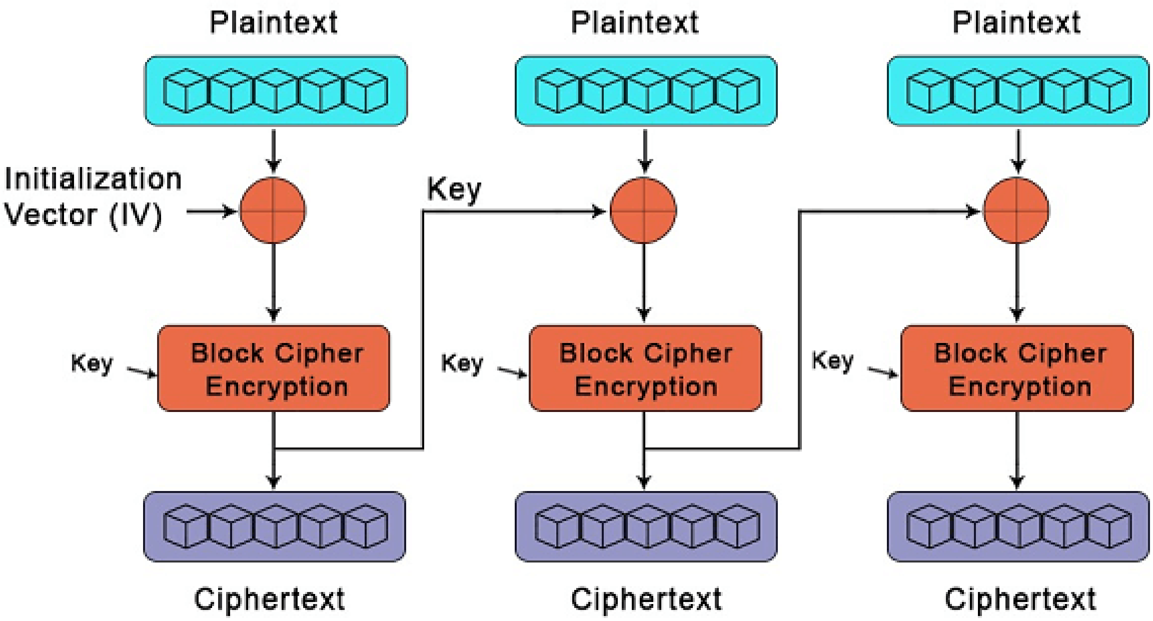
Plain Text : 10011001

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

Block Cipher:

Block cipher is an encryption algorithm that takes a fixed size of input say *b* bits and produces a ciphertext of *b* bits again.

If the input is larger than *b* bits it can be divided further.



What is an Initialization Vector (IV)?

- An IV is a random value used in combination with a block cipher to enhance security.
- It ensures that even if the same plaintext is encrypted multiple times with the same key, the resulting ciphertext will be different each time.

Why is an IV Important?

- Without an IV, encrypting the same plaintext with the same key would produce the same ciphertext. This predictability can be exploited by attackers.

- The IV adds randomness to the encryption process, making it more difficult for attackers to use brute force or other techniques to break the encryption.

Example to Illustrate

Imagine you have a simple message: "HELLO WORLD"

1. **Plaintext and Key:**
   - Plaintext: "HELLO WORLD"
   - Key: A secret key known only to the sender and receiver.
2. **Without IV:**
   - Encrypt "HELLO WORLD" with the key: Result is "ABCDEF123456"
   - Encrypt "HELLO WORLD" again with the same key: Result is still "ABCDEF123456"
   - An attacker can see that the same plaintext produces the same ciphertext every time.

3. **With IV:**

   - Generate a random IV: Let's say the IV is "XYZ123"
   - Combine the IV with the plaintext: "XYZ123" + "HELLO WORLD"
   - Encrypt the combined text with the key: Result is "MNO789456321"
   - Encrypt "HELLO WORLD" again with a different IV (e.g., "ABC987"): Result is "QWE654789012"
   - The ciphertext is different each time, even though the plaintext and key are the same.

Modes of Operation of a Block Cipher:

Electronic Code Book (ECB):

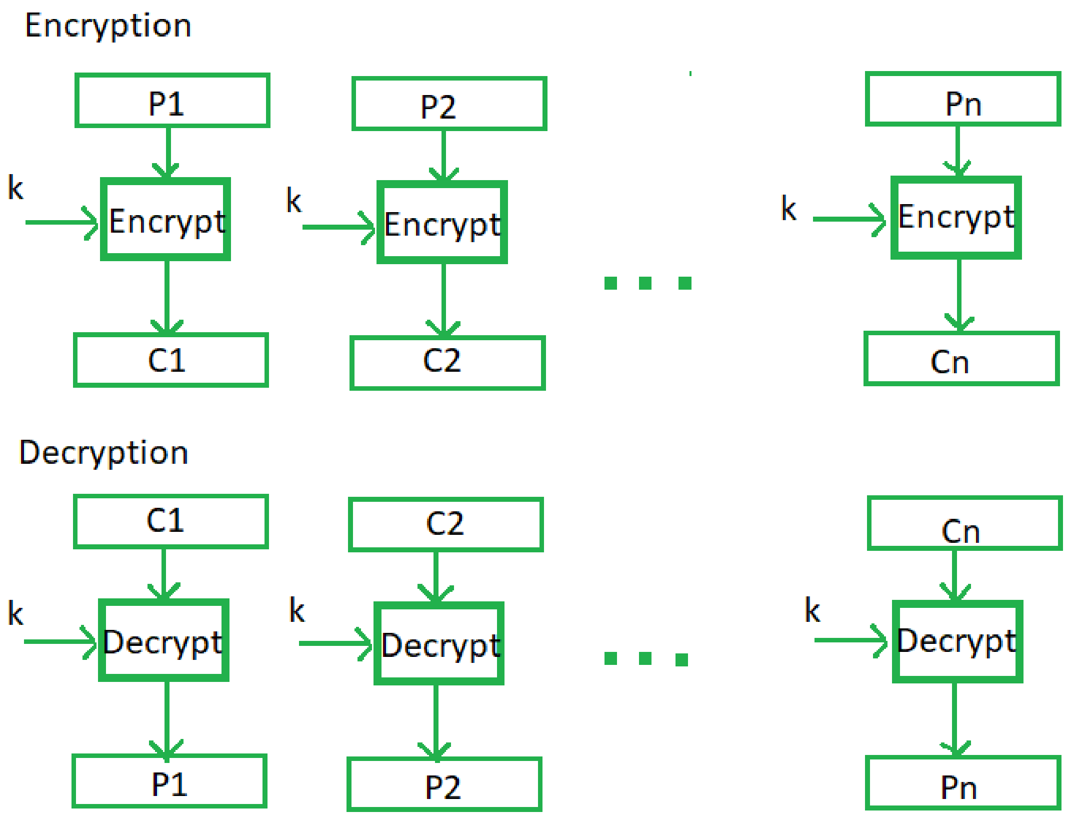
Simplest mode of operation.

Plain text is divided into a no. of fixed size block.

If message is not a multiple of block size, than padding is done

Take one block at a time and encrypt it.

Same key used for encryption and decryption.



Advantages of using ECB

- Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
- Simple way of the block cipher.

Disadvantages of using ECB

- If identical block appears then this mode produces same cipher. Example:

  Consider a plaintext message: "ATTACK AT DAWN"

  If we use a block size of 4 characters (e.g., 32 bits), the plaintext is divided into blocks:

  1. "ATTA"
  2. "CK A"
  3. "T DA"
  4. "WN.."

  Each block is then encrypted independently. If "ATTA" appears more than once in the plaintext, it will result in the same ciphertext block each time, revealing the pattern to an observer.

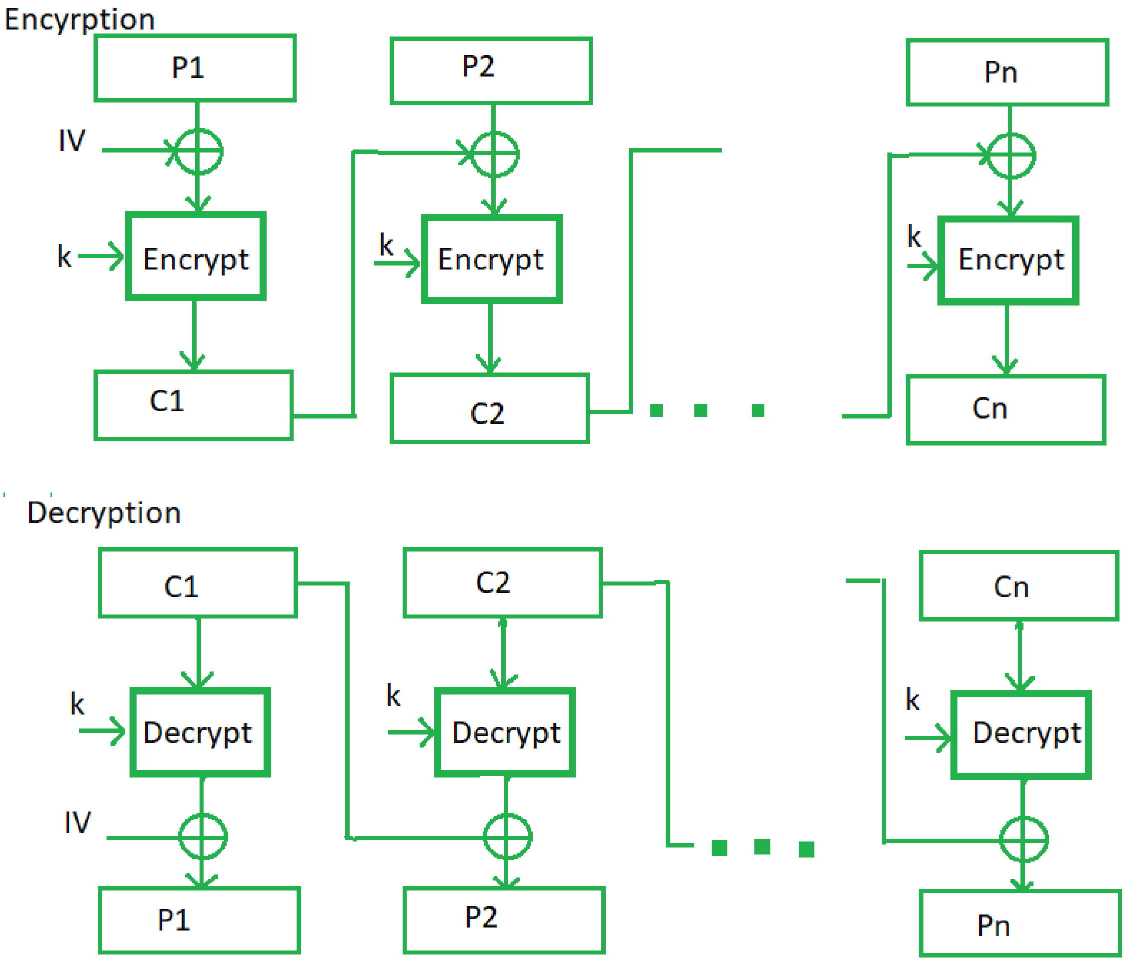
When to Use ECB

Due to its security weaknesses, ECB mode is generally recommended only for situations where:

- The data is random or already compressed.
- The data size is very small and does not contain repeating patterns.
- Security is not the primary concern (e.g., for certain types of test data or very limited use cases).

Cipher Block Chaining (CBC):

Cipher Block Chaining (CBC) is a mode of operation for block ciphers that enhances the security of encrypted data. It is designed to provide confidentiality by ensuring that identical plaintext blocks produce different ciphertext blocks, even when the same key is used.



How CBC Works

1. **Initialization Vector (IV):**
    - CBC requires an Initialization Vector (IV), which is a random value used for the first block of plaintext. The IV does not need to be secret, but it should be unique and unpredictable for each encryption session.

2. **Encryption Process:**
   - The plaintext is divided into fixed-size blocks (e.g., 128-bit blocks for AES).
   - For the first block of plaintext, the IV is XORed (bitwise exclusive OR operation) with the plaintext block before encryption.
   - The result is then encrypted using the block cipher and a symmetric key, producing the first ciphertext block.
   - For subsequent blocks, the previous ciphertext block is XORed with the current plaintext block before encryption. This chaining process continues for all plaintext blocks.
3. **Decryption Process:**
   - The ciphertext is divided into the same block size as used during encryption.
   - The first ciphertext block is decrypted using the block cipher and the symmetric key.
   - The resulting decrypted block is then XORed with the IV to retrieve the first plaintext block.

   - For subsequent blocks, the current ciphertext block is decrypted, and the result is XORed with the previous ciphertext block to retrieve the plaintext block. This process continues for all ciphertext blocks.

Encryption Example:

1. Divide the plaintext into blocks:
   - Plaintext: P1, P2, P3, ...
2. Generate a random IV: IV
3. Encrypt each block:
   - C1 = E_K(P1 XOR IV) (where E_K denotes encryption with key K)
   - C2 = E_K(P2 XOR C1)
   - C3 = E_K(P3 XOR C2)
   - And so on...

Decryption Example:

1. Divide the ciphertext into blocks:
   - Ciphertext: C1, C2, C3, ...

2. Decrypt each block:
    - P1 = D_K(C1) XOR IV (where D_K denotes decryption with key K)
    - P2 = D_K(C2) XOR C1
    - P3 = D_K(C3) XOR C2
    - And so on...

Advantages:

- CBC works well for input greater than *b* bits.
- CBC is a good authentication mechanism.

Disadvantages:

- Errors in transmission can affect multiple blocks, complicating error correction.
- Parallel encryption is not possible since every encryption requires a previous cipher.

Other modes:

- Cipher Feedback Mode (CFB)
- Output Feedback Mode (OFB)
- Counter Mode

Examples of Block Ciphers:

Data Encryption Standard (DES)

- **Overview**: Developed in the 1970s, DES was the first encryption standard adopted by the U.S. government. It operates on 64-bit blocks of plaintext using a 56-bit symmetric key.
- **Usage**: Initially widely used for secure communications and data encryption, DES later faced criticism due to its short key length and vulnerability to brute-force attacks.
- **Discontinuation**: DES was officially discontinued by the National Institute of Standards and Technology (NIST) in 2005, primarily due to security concerns.

Advanced Encryption Standard (AES)

- **Overview**: AES is a successor to DES and was established as a standard by the U.S. National Institute of Standards and Technology (NIST) in 2001. It operates on 128-bit blocks of plaintext using symmetric keys of 128, 192, or 256 bits.
- **Strengths**: AES is highly regarded for its security and efficiency. Its underlying algorithm employs substitution-permutation and diffusion techniques to provide strong encryption.
- **Adoption**: AES has become the de facto encryption standard worldwide and is extensively used in various applications, including secure communications, data storage, and financial transactions.

Block Cipher	**Stream Cipher**
Block Cipher Converts the plain text into cipher text by taking plain text’s block at a time.	Stream Cipher Converts the plain text into cipher text by taking 1 bit plain text at a time.
Block cipher uses either 64 bits or more than 64 bits.	While stream cipher uses 8 bits.
The complexity of block cipher is simple.	While stream cipher is more complex.
Block cipher Uses confusion as well as diffusion.	While stream cipher uses only confusion.
In block cipher, reverse encrypted text is hard.	While in-stream cipher, reverse encrypted text is easy.

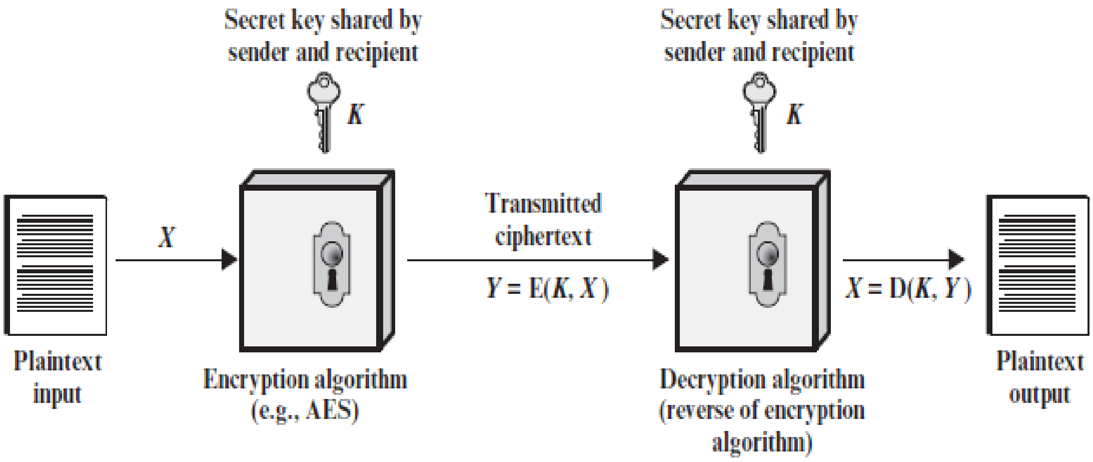
Block Cipher	**Stream Cipher**
The algorithm modes which are used in block cipher are ECB (Electronic Code Book) and CBC (Cipher Block Chaining).	The algorithm modes which are used in stream cipher are CFB (Cipher Feedback) and OFB (Output Feedback).
Block cipher is slow as compared to a stream cipher.	While stream cipher is fast in compariso to block cipher.
More secure than stream ciphers when the same key is used multiple times.	Less secure than block ciphers when the same key is used multiple times.
key length is typically 128 or 256 bits.	key length is typically 128 or 256 bits.
Operates on fixed-length blocks of data.	Encrypts data one bit at a time.

Symmetric vs. Asymmetric Ciphers:

Symmetric Ciphers:

It uses a single key to encrypt and decrypt information.

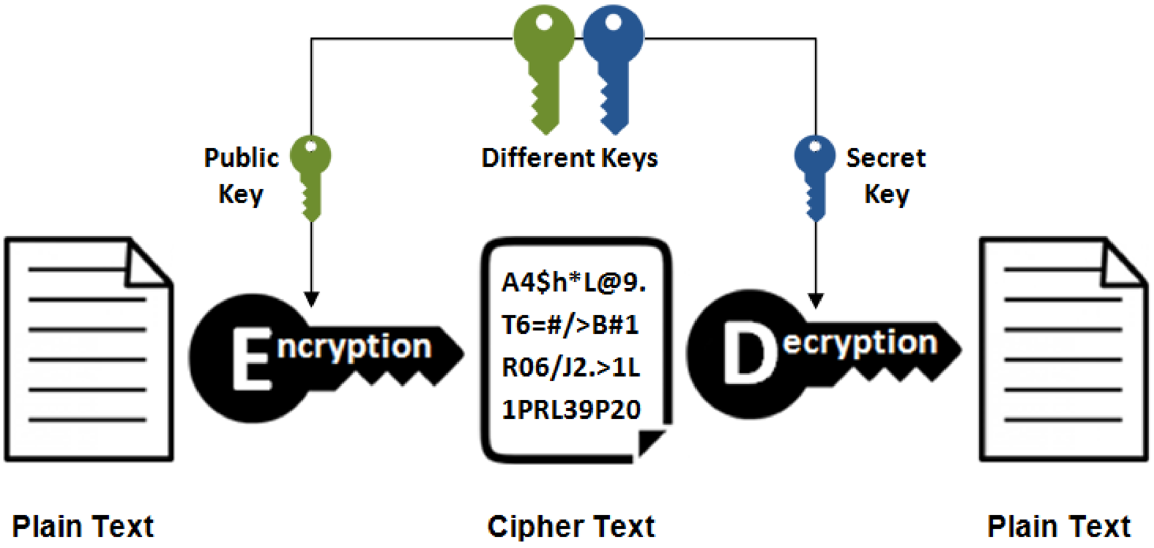
This key must be kept secret to ensure the security of the encrypted information.


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D(K, Y)
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

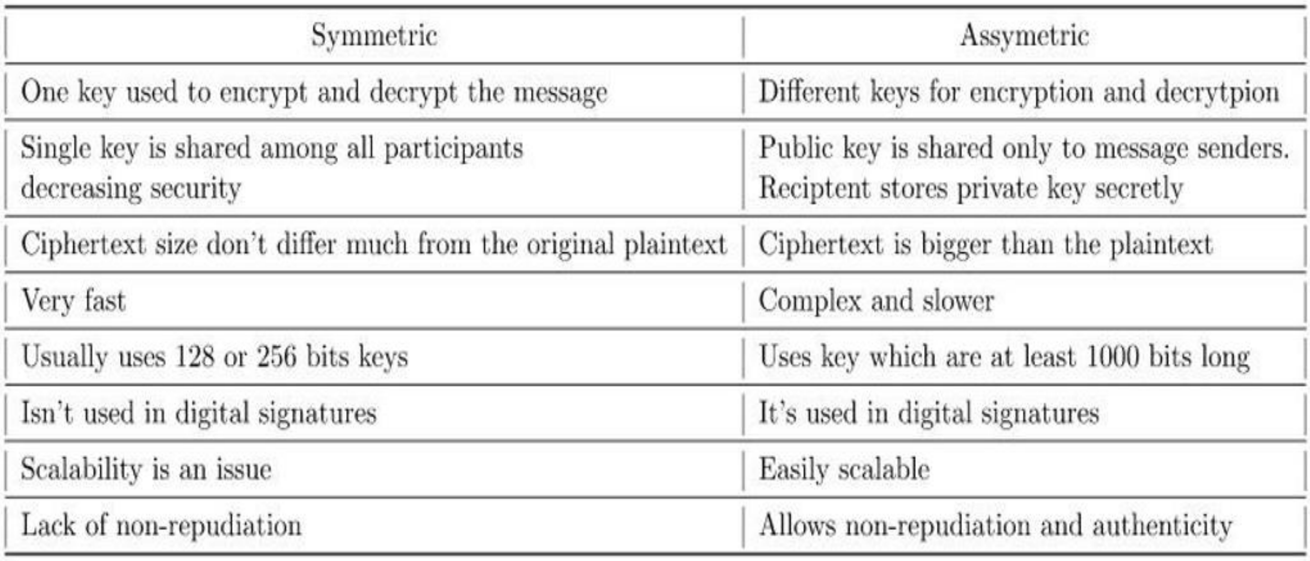
Asymmetric Cipher:

It uses two keys, a public key and a private key, to encrypt and decrypt information.

The public key can be freely distributed, but the private key must be kept secret.



Symmetric vs. Asymmetric Ciphers:



Symmetric	Assymetric
One key used to encrypt and decrypt the message	Different keys for encryption and decrytpion
Single key is shared among all participants decreasing security	Public key is shared only to message senders. Reciptent stores private key secretly
Ciphertext size don't differ much from the original plaintext	Ciphertext is bigger than the plaintext
Very fast	Complex and slower
Usually uses 128 or 256 bits keys	Uses key which are at least 1000 bits long
Isn't used in digital signatures	It's used in digital signatures
Scalability is an issue	Easily scalable
Lack of non-repudiation	Allows non-repudiation and authenticity

Symmetric Encryption:

Feistel Cipher Structure:

Feistel Cipher model is a structure or a design used to develop many block ciphers such as DES.

Same encryption as well as decryption algorithm is used. A separate key is used for each round.

However same round keys are used for encryption as well as decryption.

How It Works

1. **Input Data and Key**:
   - **Plaintext Block**: This is the data you want to encrypt, split into two equal parts (L0 and R0).
   - **Key (K)**: A secret key used to encrypt the data.
2. **Processing Rounds**:
   - The two parts (L0 and R0) go through several rounds of processing (usually 16 rounds).

3. **Each Round**:
   - **Inputs**: The two parts from the previous round (Li-1 and Ri-1) and a unique subkey (Ki) derived from the main key (K).
   - **Subkey Generation**: Subkeys (Ki) are different for each round and are generated from the main key.

Round Details

1. **Substitution**:
   - Substitution is like replacing parts of the data with something else.
   - **Function F**: Apply a function F to the right half (Ri-1).
   - **XOR Operation**: XOR the output of F with the left half (Li-1). XOR is a bitwise operation that combines two binary values.
   - **Example**: If Li-1 is 1010 and the output of F(Ri-1) is 1100, XORing them gives 0110.

2. **Permutation**:
   - Permutation is like rearranging the order of the data.
   - **Swap Halves**: Swap the left and right halves. After the swap, the new left half is the old right half, and the new right half is the result of the XOR operation.

Example Round

- Start with L0 and R0.
- Compute F(R0).
- XOR the result of F(R0) with L0 to get the new right half.
- Swap L0 and the new right half to get L1 and R1 for the next round.

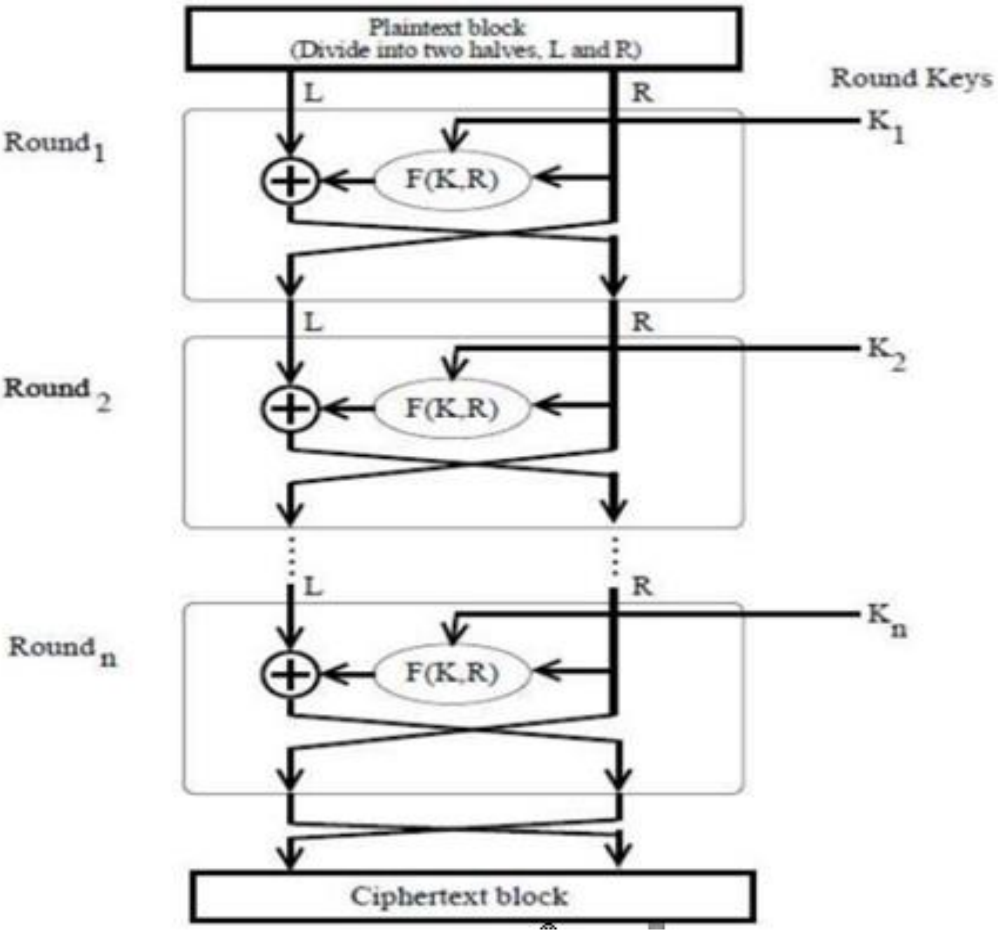
Final Output

- After n rounds, combine the final left and right halves to get the ciphertext, which is the encrypted version of the original data.

Key Points

- **Block Size**: This is the size of the data block. 128 bits is common as it offers good security and speed.

- **Key Size**: Larger keys are more secure. 128-bit keys are standard in modern encryption.
- **Number of Rounds**: More rounds mean better security. Typically, 16 rounds are used.
- **Subkey Generation**: The process of creating different subkeys from the main key. More complexity here means harder to break.
- **Round Function (F)**: The function used in substitution. More complex functions improve security.



Feistel Cipher Decryption:

Decryption with a symmetric block cipher is essentially the same as the encryption process.

The rule is as follows: Use the ciphertext as input to the algorithm, but use the subkeys *Ki* in reverse order.

That is, use *Kn* in the first round, *Kn*-1 in the second round, and so on until *K*1 is used in the last round.

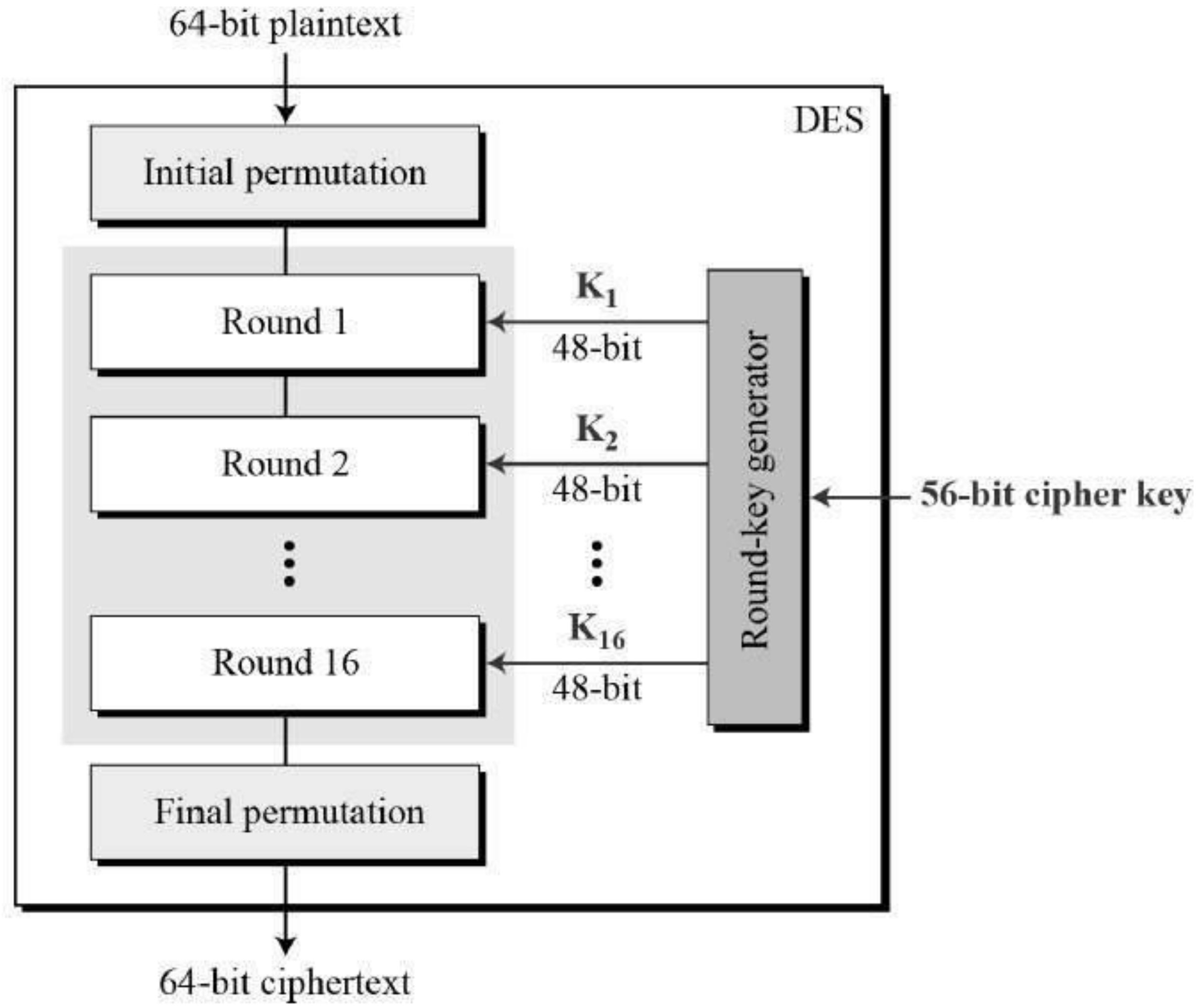
This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption.

Data Encryption Standard (DES):

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

General Structure of DES is depicted in the following illustration −



how the DES algorithm works:

1. **Key Generation**: The 56-bit key is expanded into 16 48-bit subkeys, one for each round of encryption/decryption. Each subkey is derived from the original key using a process called key scheduling, which involves permutation and rotation operations.

2. **Initial Permutation (IP)**: The 64-bit plaintext is permuted according to a fixed permutation table. This initial permutation reorders the bits of the plaintext.
3. **Feistel Rounds**: The permuted plaintext is divided into two 32-bit halves, left and right. The right half is expanded to 48 bits using another fixed permutation table. Then, each round consists of the following steps:
   - The expanded right half is XORed with the current round's subkey.
   - The result is passed through a series of S-boxes (substitution boxes), which perform a non-linear substitution operation.
   - The output from the S-boxes is permuted using a fixed permutation table.
   - The permuted output is XORed with the left half.
   - The left and right halves are swapped.
4. **Final Permutation (FP)**: After 16 rounds of processing, the left and right halves are combined and passed through a final permutation, which is the inverse of the initial permutation.

5. **Decryption**: Decryption with DES is essentially the same process as encryption, but with the subkeys used in reverse order. The ciphertext is processed through the same Feistel network structure, with the subkeys applied in reverse order from K16 to K1.

Basic Concepts of Fields:

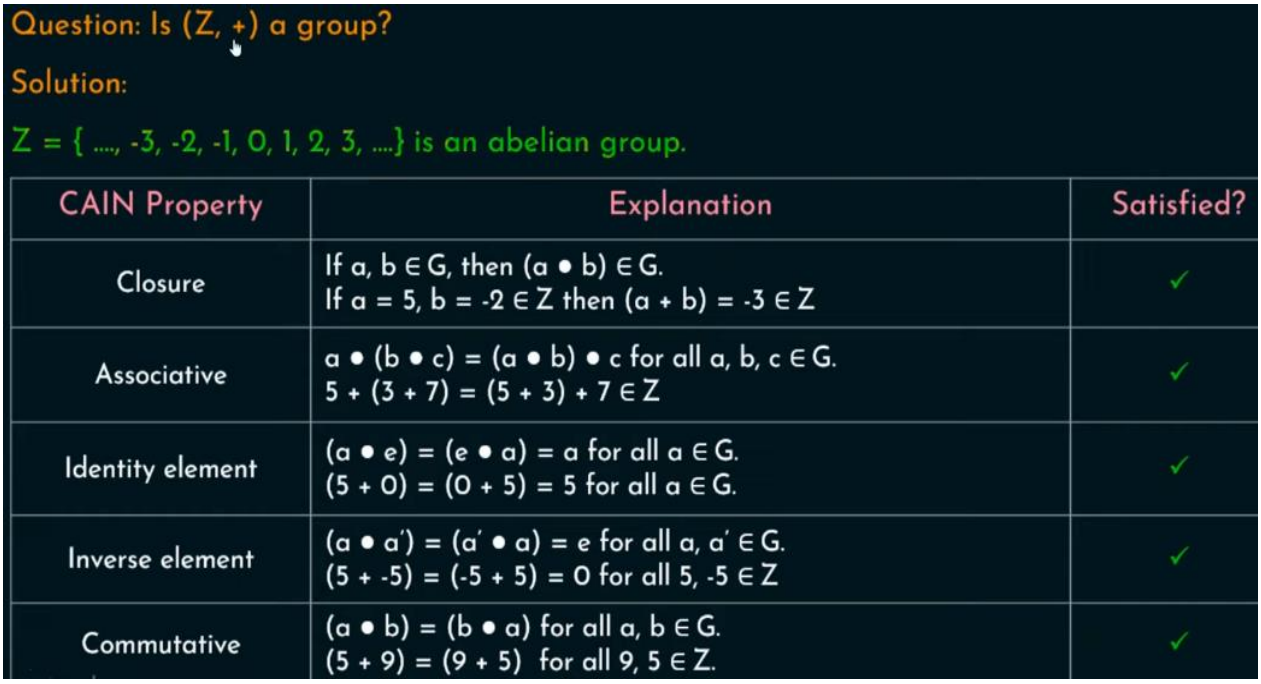
Groups:

A group G, sometimes denoted by {G,.}, is set of elements with binary operations denoted by # that associates to each ordered pair (a,b) of elements in G an element (a.b) in G,such that following axioms are satisfied.

1. Closure: -> If a and b belongs to G, then a.b is also in G
2. Associative:-> a.(b.c)=(a.b).c for all a,b,c in G
3. Identity Element: -> There is an element e in G such that a.e = e.a =a for all a in G
4. Inverse Element: -> For each a in G, there is an element a' in G such that a.a'=a'.a=e

A group is called abelian if it satisfies the following additional condition

- Commutative: a.b = b.a for all a,b in G



Question: Is (Z, +) a group?

Solution:

Z = { ...., -3, -2, -1, 0, 1, 2, 3, ...} is an abelian group.

CAIN Property	Explanation	Satisfied?
Closure	If a, b ∈ G, then (a • b) ∈ G. If a = 5, b = -2 ∈ Z then (a + b) = -3 ∈ Z	✓
Associative	a • (b • c) = (a • b) • c for all a, b, c ∈ G. 5 + (3 + 7) = (5 + 3) + 7 ∈ Z	✓
Identity element	(a • e) = (e • a) = a for all a ∈ G. (5 + 0) = (0 + 5) = 5 for all a ∈ G.	✓
Inverse element	(a • a′) = (a′ • a) = e for all a, a′ ∈ G. (5 + -5) = (-5 + 5) = 0 for all 5, -5 ∈ Z	✓
Commutative	(a • b) = (b • a) for all a, b ∈ G. (5 + 9) = (9 + 5) for all 9, 5 ∈ Z.	✓

Ring:

A ring R denoted by {R, +, *}, is a set of elements with two binary operations, called addition and multiplications , such that for all a,b,c

- Group (A1-A4), Abelian Group(A5)

- Closure under multiplications (M1): if a,b ∈ R then ab ∈ R
- Associative of multiplications (M2) a(bc) = (ab)c for all a,b,c ∈ R
- Distributive laws (M3)
    - a(b+c) = ab + ac for all a,b,c ∈ R
    - (a+b) c = ac + bc for all a,b,c ∈ R

Commutative Rings: A ring is said to be commutative, if it satisfies the following additional condition:

- Commutative of multiplication (M4): ab = ba for all a,b ∈ R

Integral Domain:

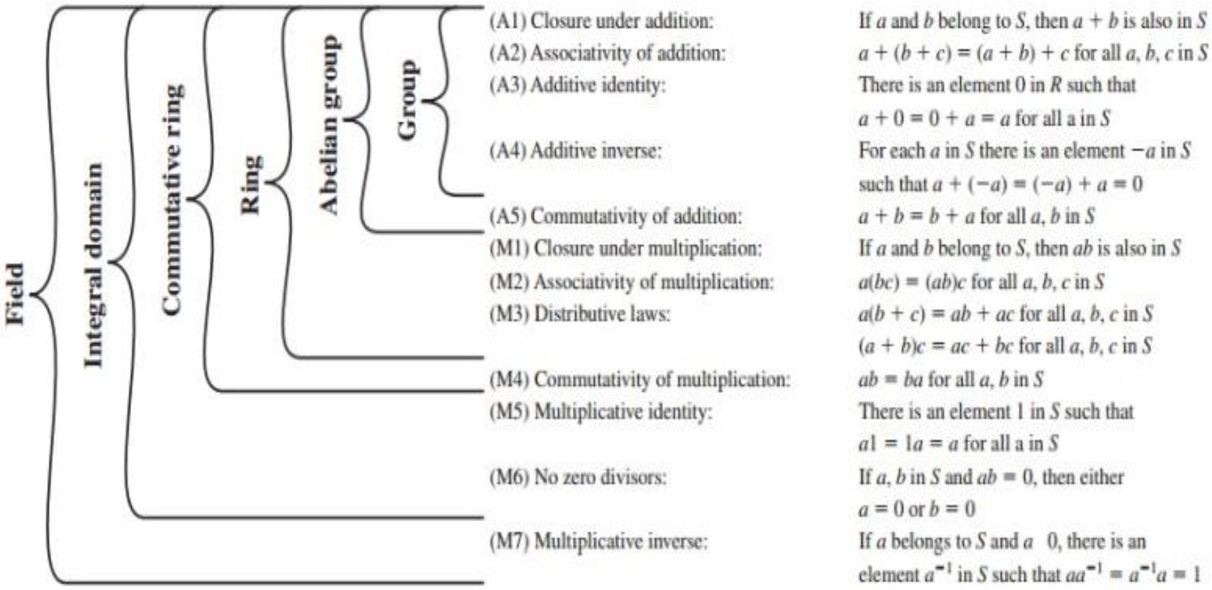
An integral domain is a commutative ring that obeys the following axioms:

- Multiplicative identity (M5): there is an element 1 ∈ R such that a1=1a=a for all a ∈R
- No Zero divisor (M6): if a,b ∈ R and ab=0, then either a=0 or b=0

Fields:

A fields F, sometimes denoted by {F,+,*}, is a set of elements with two binary operations, called addition and multiplication, such that for all a,b,c ∈ F, the following axioms are satisfied

- (A1-M6): F is an integral domain; that is, F satisfies axioms A1-A5 and M1-M6.
- Multiplicative inverse (M7): For each a in F, except 0, there is an element a' in F such that aa'=(a')a =1



(A1) Closure under addition: If a and b belong to S, then $a + b$ is also in S

(A2) Associativity of addition: $a + (b + c) = (a + b) + c$ for all a, b, c in S

(A3) Additive identity: There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S

(A4) Additive inverse: For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$

(A5) Commutativity of addition: $a + b = b + a$ for all a, b in S

(M1) Closure under multiplication: If a and b belong to S, then ab is also in S

(M2) Associativity of multiplication: $a(bc) = (ab)c$ for all a, b, c in S

(M3) Distributive laws: $a(b + c) = ab + ac$ for all a, b, c in S
$(a + b)c = ac + bc$ for all a, b, c in S

(M4) Commutativity of multiplication: $ab = ba$ for all a, b in S

(M5) Multiplicative identity: There is an element 1 in S such that $a1 = 1a = a$ for all a in S

(M6) No zero divisors: If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$

(M7) Multiplicative inverse: If a belongs to S and a 0, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Figure: Groups, Rings, and Field

Galois Fields:

Galois Fields, named after the mathematician Évariste Galois, are sets of numbers with a finite number of elements that follow specific rules for addition and multiplication.

These rules ensure that any operation performed within the set results in another element from the same set, making the field "closed."

Galois Fields are used in areas like cryptography, coding theory, and error correction because of their unique properties.

What is a Galois Field?

A Galois Field (GF) is a finite set of numbers with defined addition and multiplication, and their respective inverses.

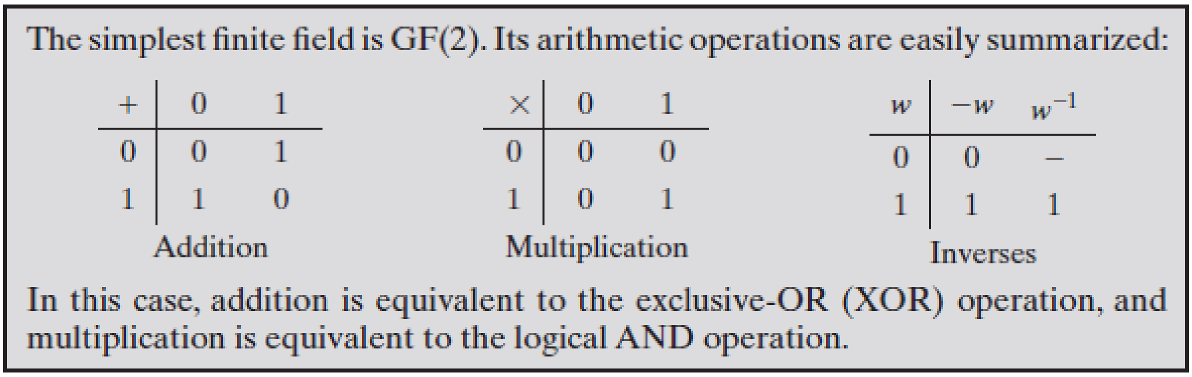
The size of a Galois Field is represented by a prime number p and is denoted as GF(p).

Example of GF(3)

Now, let's consider GF(3), which has three elements: 0, 1, and 2.

- **Addition in GF(3)**: Addition is performed modulo 3.

  - 0+0=0
  - 0+1=1
  - 0+2=2
  - 1+1=2
  - 1+2=3 (which is 0 in GF(3) because 3 modulo 3 is 0)
  - 2+2=4 (which is 1 in GF(3) because 4 modulo 3 is 1)
- **Multiplication in GF(3)**: Multiplication is also performed modulo 3.
  - 0×0=0
  - 0×1=0
  - 0×2=0
  - 1×1=1
  - 1×2=2
  - 2×2=4 (which is 1 in GF(3) because 4 modulo 3 is 1)



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

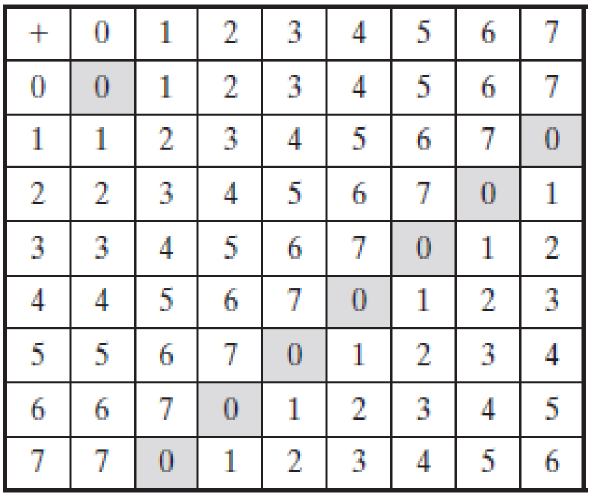
Multiplication

w	$-w$	w^{-1}
0	0	–
1	1	1

Inverses

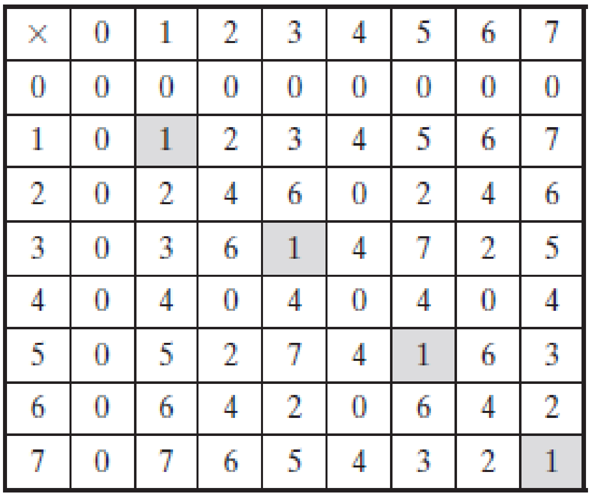
In this case, addition is equivalent to the exclusive-OR (XOR) operation, and multiplication is equivalent to the logical AND operation.

Arithmetic Modulo 8 and Modulo 7



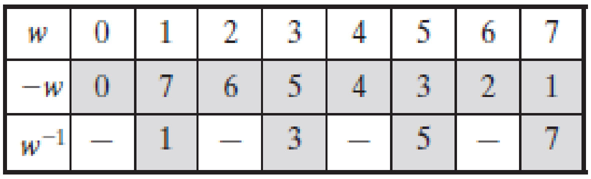
+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8



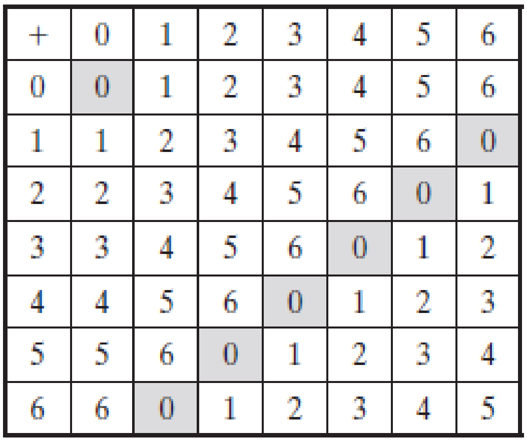
×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8



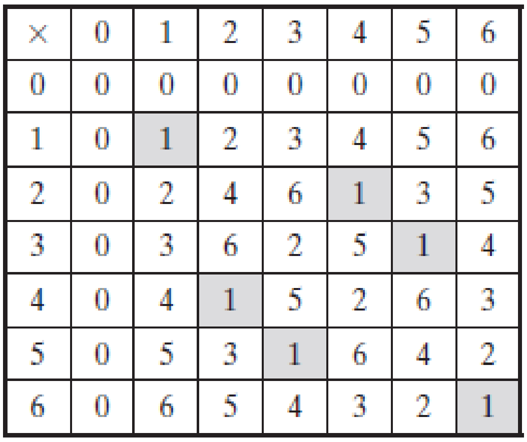
w	0	1	2	3	4	5	6	7
$-w$	0	7	6	5	4	3	2	1
w^{-1}	—	1	—	3	—	5	—	7

(c) Additive and multiplicative inverses modulo 8



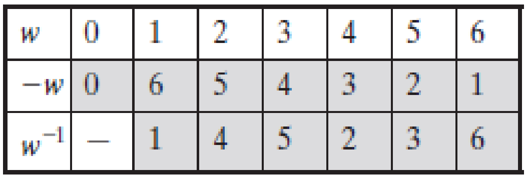
+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(d) Addition 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

(e) Multiplication modulo 7



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

(f) Additive and multiplicative inverses modulo 7

Modular Arithmetic:

Modular arithmetic, also known as clock arithmetic or arithmetic modulo n, is a system of arithmetic for integers where numbers "wrap around" after reaching a certain modulus n.

It is often denoted by the symbol mod or % in programming languages.

Modular arithmetic enables us to simply make groups, rings and fields which are the basic constructing piece of most modern public-key cryptosystems

Theorem − n is an equivalence relation on the integers. An equivalence class includes those integers which have the equal remainder on division by n.

The equivalence classes are also called a congruence classes modulo n.

Instead of say the integers a and b are equivalent and it can said that they are congruent modulo n.

The set of all integers congruent to a modulo n is called the residue class [a].

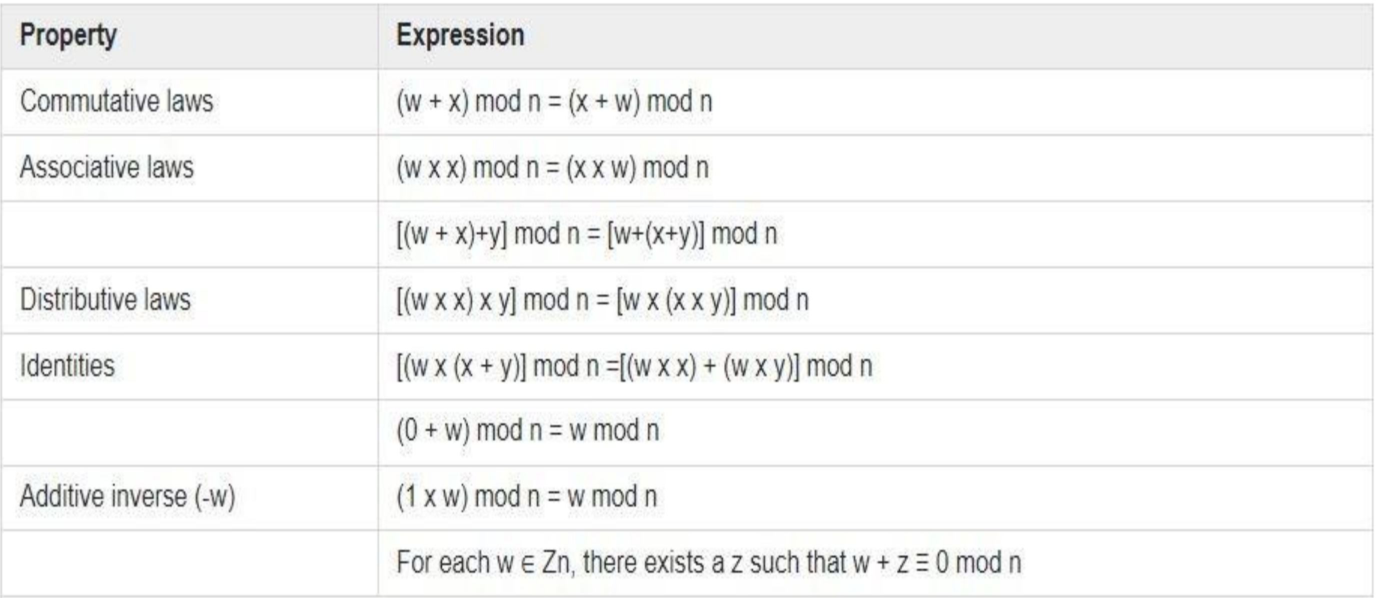
The modulo operator has the following properties −

- a ≡ b mod n if n|(a − b).
- (a mod n) = (b mod n) implies a ≡ b mod n.
- a ≡ b mod n implies b ≡ a mod n.
- a ≡ b mod n and b ≡ c mod n imply a ≡ c mod n.

Properties of modular arithmetic operations

- [(a mod n) + (b mod n)] mod n = (a + b) mod n
- [(a mod n) - (b mod n)] mod n = (a - b) mod n
- [(a mod n) x (b mod n)] mod n = (a x b) mod n

Let Zn = {0, 1, 2,… (n-1)}, be the set of residues modulus n.



Property	Expression
Commutative laws	(w + x) mod n = (x + w) mod n
Associative laws	(w x x) mod n = (x x w) mod n
	[(w + x)+y] mod n = [w+(x+y)] mod n
Distributive laws	[(w x x) x y] mod n = [w x (x x y)] mod n
Identities	[(w x (x + y)] mod n =[(w x x) + (w x y)] mod n
	(0 + w) mod n = w mod n
Additive inverse (-w)	(1 x w) mod n = w mod n
	For each w ∊ Zn, there exists a z such that w + z ≡ 0 mod n

Advanced Encryption Standards (AES):

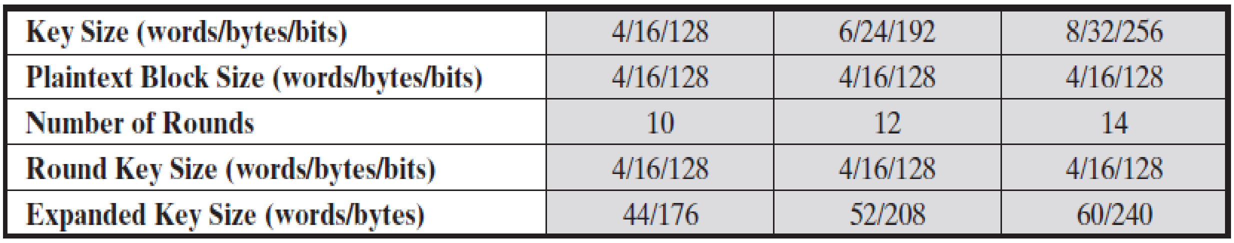
Advanced Encryption Standards is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001.

AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Points to remember

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each i.e., 16 bytes =4 word.

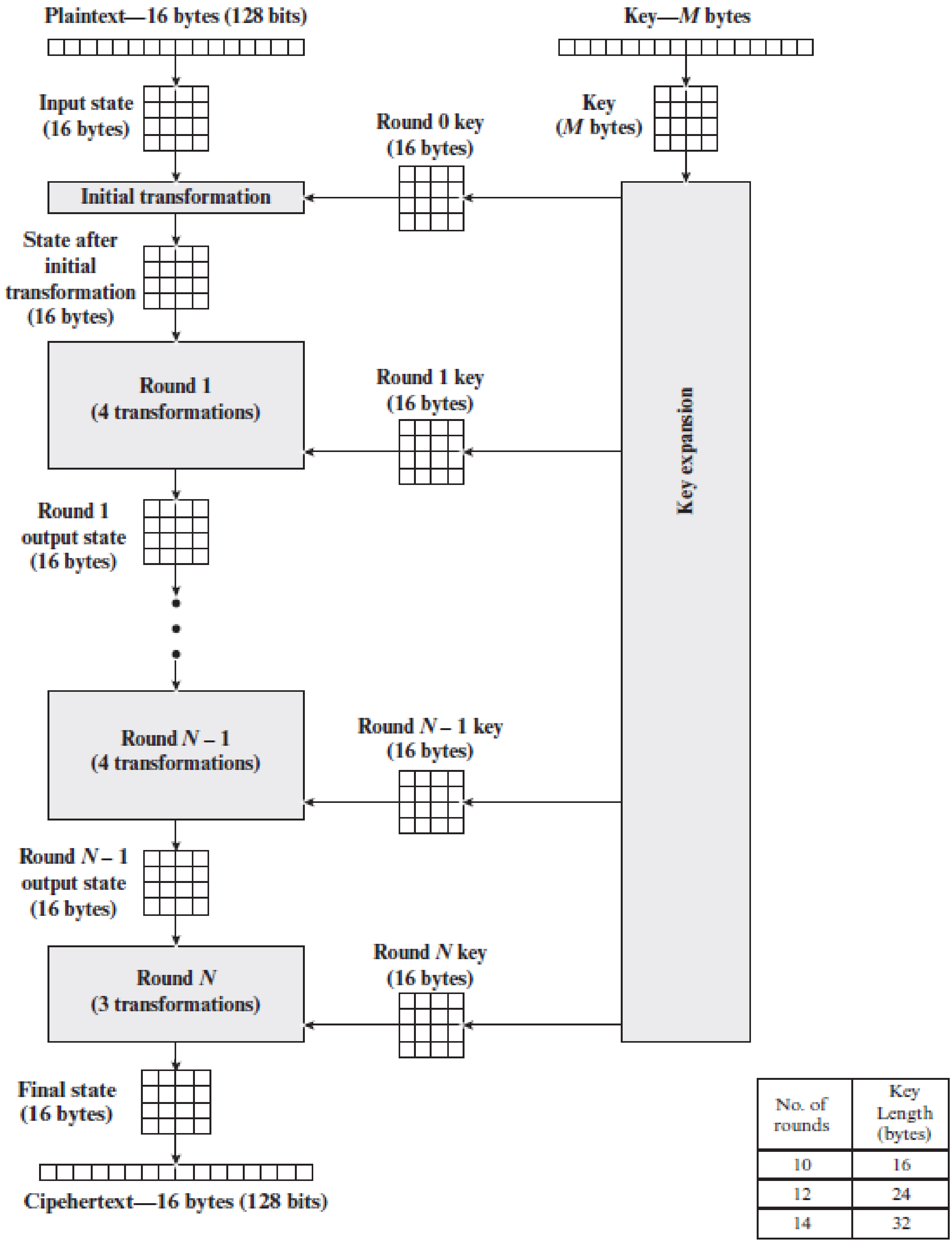
AES Parameters:



Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

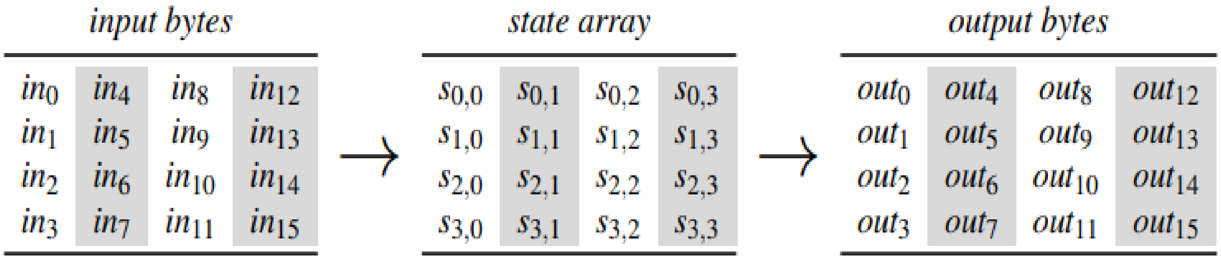
Note:

Number of key generated by key expansion = (no. of round +1)


Plaintext—16 bytes (128 bits)
Key—M bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Key expansion
Round 1 (4 transformations)
Round 1 key (16 bytes)
Round 1 output state (16 bytes)
Round N – 1 (4 transformations)
Round N – 1 key (16 bytes)
Round N – 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure: AES Encryption Process



input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

state array

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

→

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Figure: State array input and output

Arrays of Words:

A word is a sequence of four bytes; a block consists of four words.

The four columns of state array s are interpreted as an array v of four words as follows, in the notation of Fig.

$$v_0 = \begin{pmatrix} s_{0,0} \\ s_{1,0} \\ s_{2,0} \\ s_{3,0} \end{pmatrix}, \quad v_1 = \begin{pmatrix} s_{0,1} \\ s_{1,1} \\ s_{2,1} \\ s_{3,1} \end{pmatrix}, \quad v_2 = \begin{pmatrix} s_{0,2} \\ s_{1,2} \\ s_{2,2} \\ s_{3,2} \end{pmatrix}, \quad v_3 = \begin{pmatrix} s_{0,3} \\ s_{1,3} \\ s_{2,3} \\ s_{3,3} \end{pmatrix}$$

Note:

(S0,0) = 0^{th} byte of 0^{th} word

(S1,2) = 1^{st} byte of 2^{nd} word

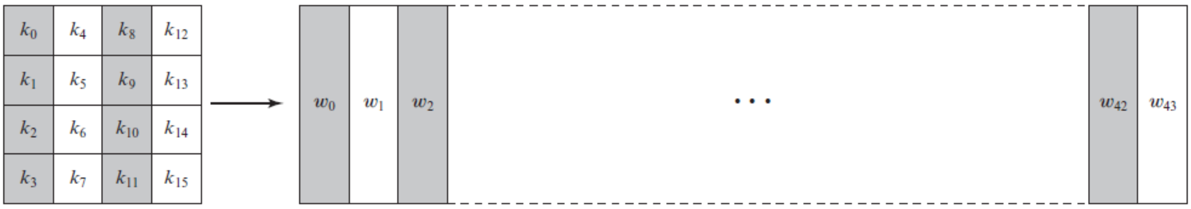
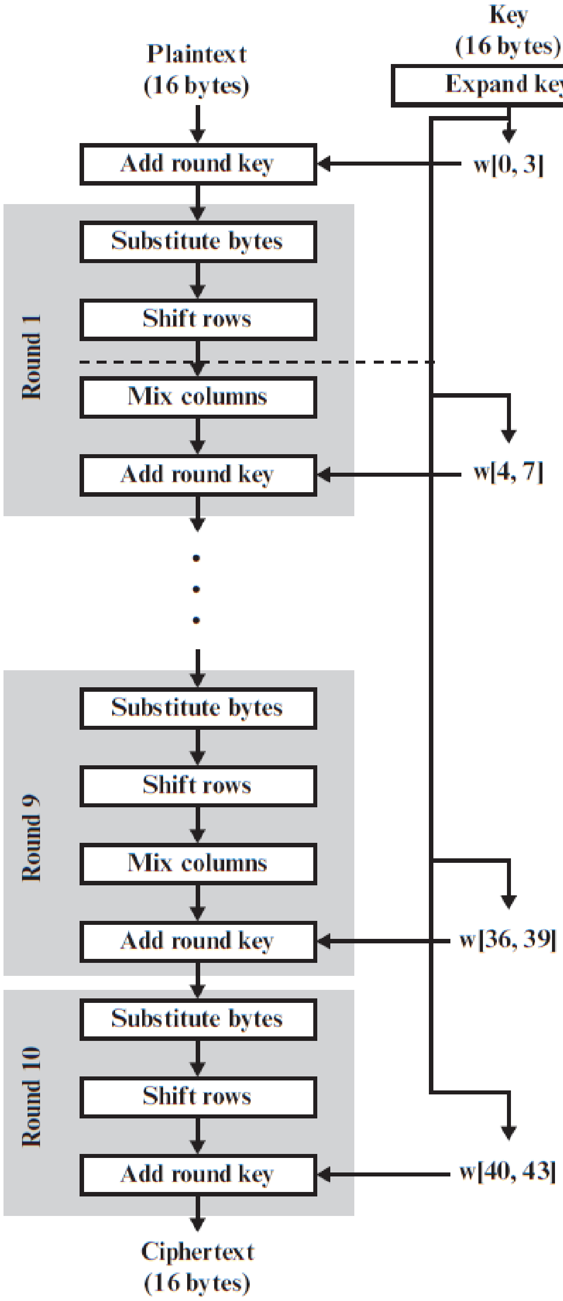


Figure: Key and expanded key



1. AES processes the entire data block as a single matrix during each round using substitutions and permutation.

2. The key that is provided as input is expanded into an array of forty-four 32-bit words, **w**[*i*]. Four distinct words (128 bits) serve as a round key for each round; these are indicated in above Figure.

3. Four different stages are used, one of permutation and three of substitution:

- **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block.
- **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.

4. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.

Four different stages:

Substitution (or Sub) Bytes:

Substitution is done for each bytes.

Only one table is used for transformation of bytes, which means that if 2 bytes are same, the transformation is also same.

At encryption side we interpret the byte as 2 hexadecimal digits.

1st hexadecimal digit = row of the substitution table

2nd hexadecimal digit = column of the substitution table

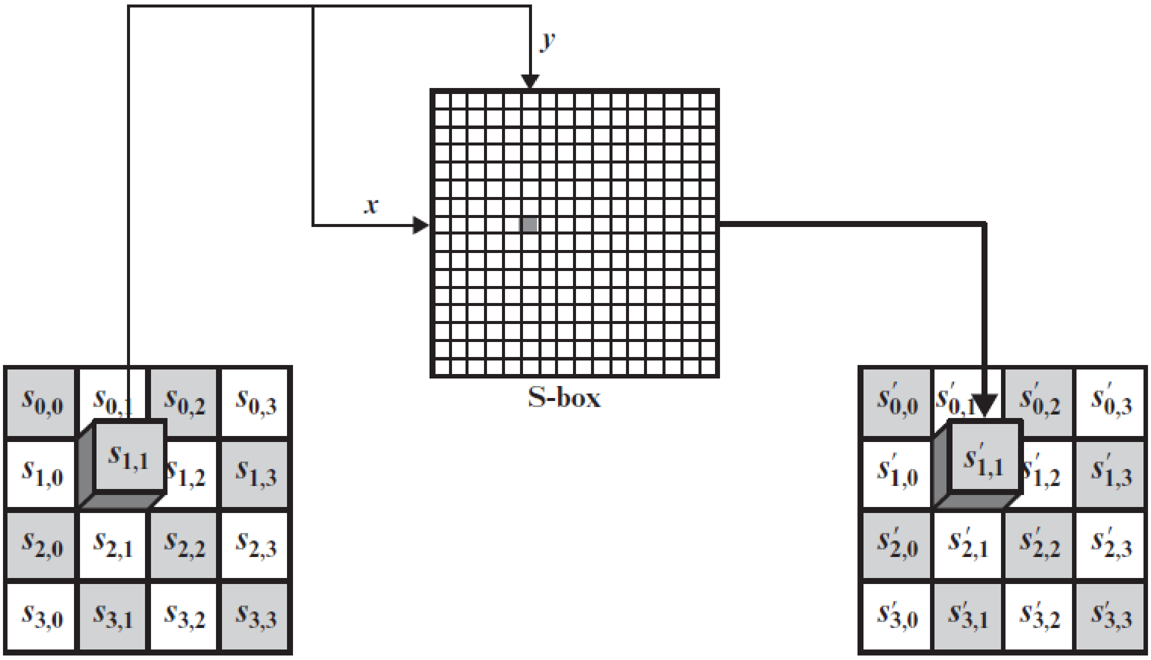
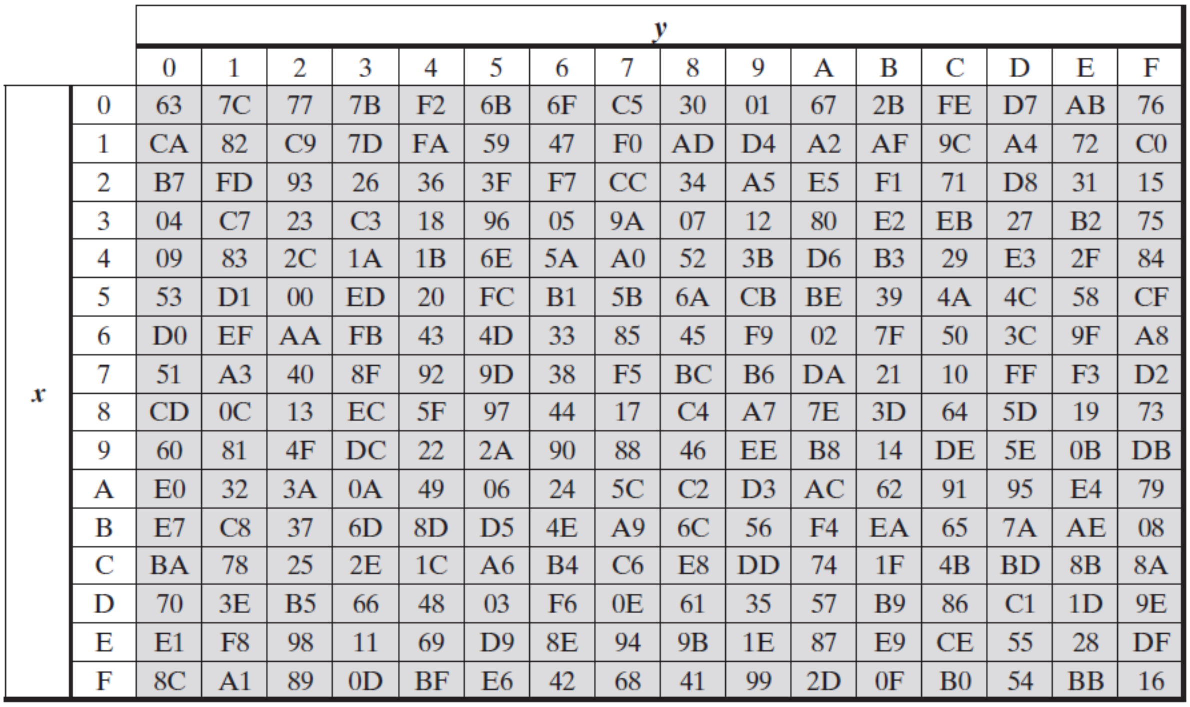


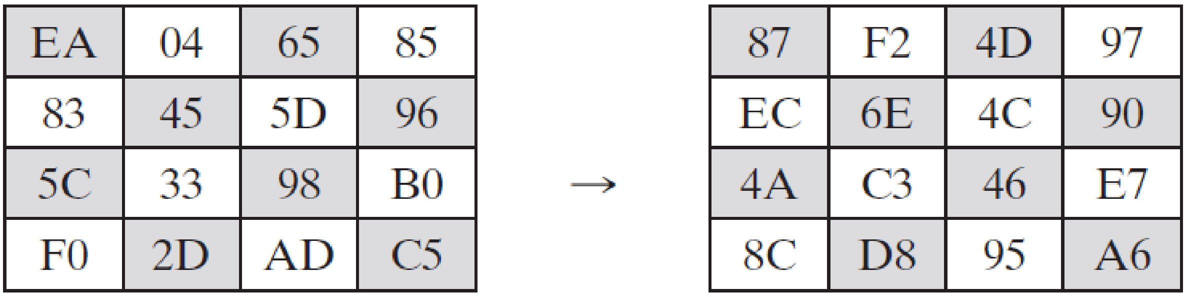
Figure: Substitute byte transformation



		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

Figure: S-box

Here is an example of the SubBytes transformation:



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

→

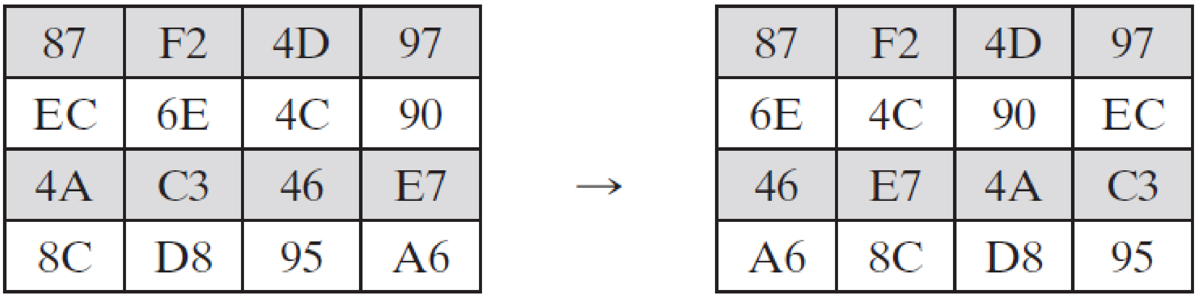
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

ShiftRows:

Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

The following is an example of ShiftRows:



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

→

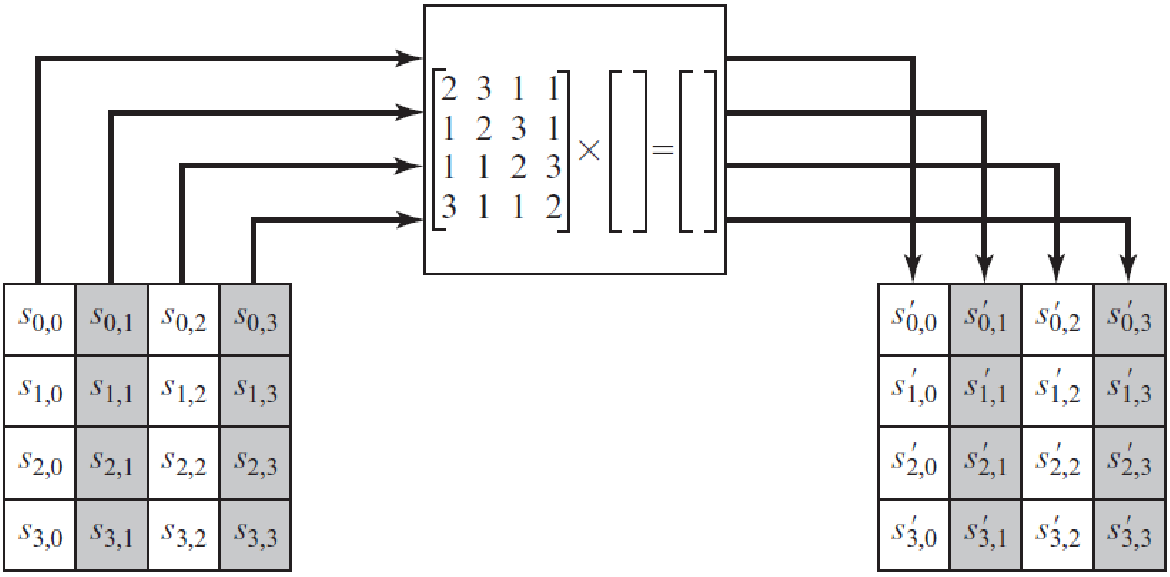
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

In Decryption, we use Inverse ShiftRows (Shifting is to the right) with number of shift is same.

Mix columns transformation:

This step is basically a matrix multiplication.

Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

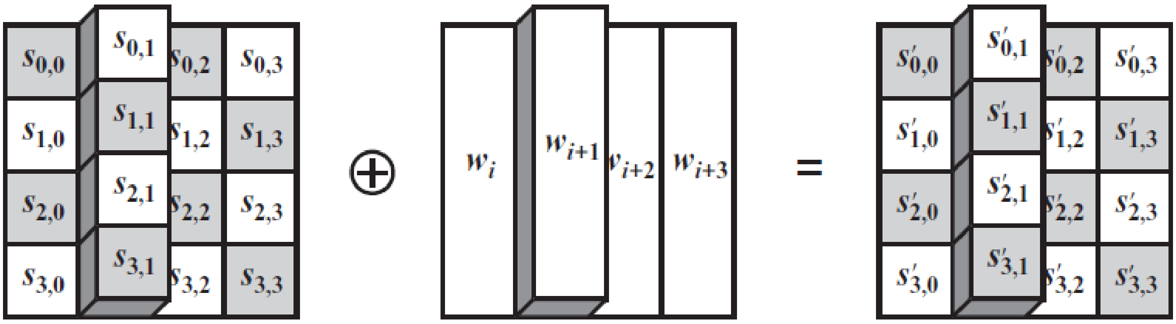


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure: Mix columns transformation

AddRoundKey Transformation:

The 128 bits of State are bitwise XORed with the 128 bits of the round key.



$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
⊕
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

$$\begin{bmatrix}47&40&A3&4C\\37&D4&70&9F\\94&E4&3A&42\\ED&A5&A6&BC\end{bmatrix} \oplus \begin{bmatrix}AC&19&28&57\\77&FA&D1&5C\\66&DC&29&00\\F3&21&41&6A\end{bmatrix} = \begin{bmatrix}EB&59&8B&1B\\40&2E&A1&C3\\F2&38&13&42\\1E&84&E7&D6\end{bmatrix}$$

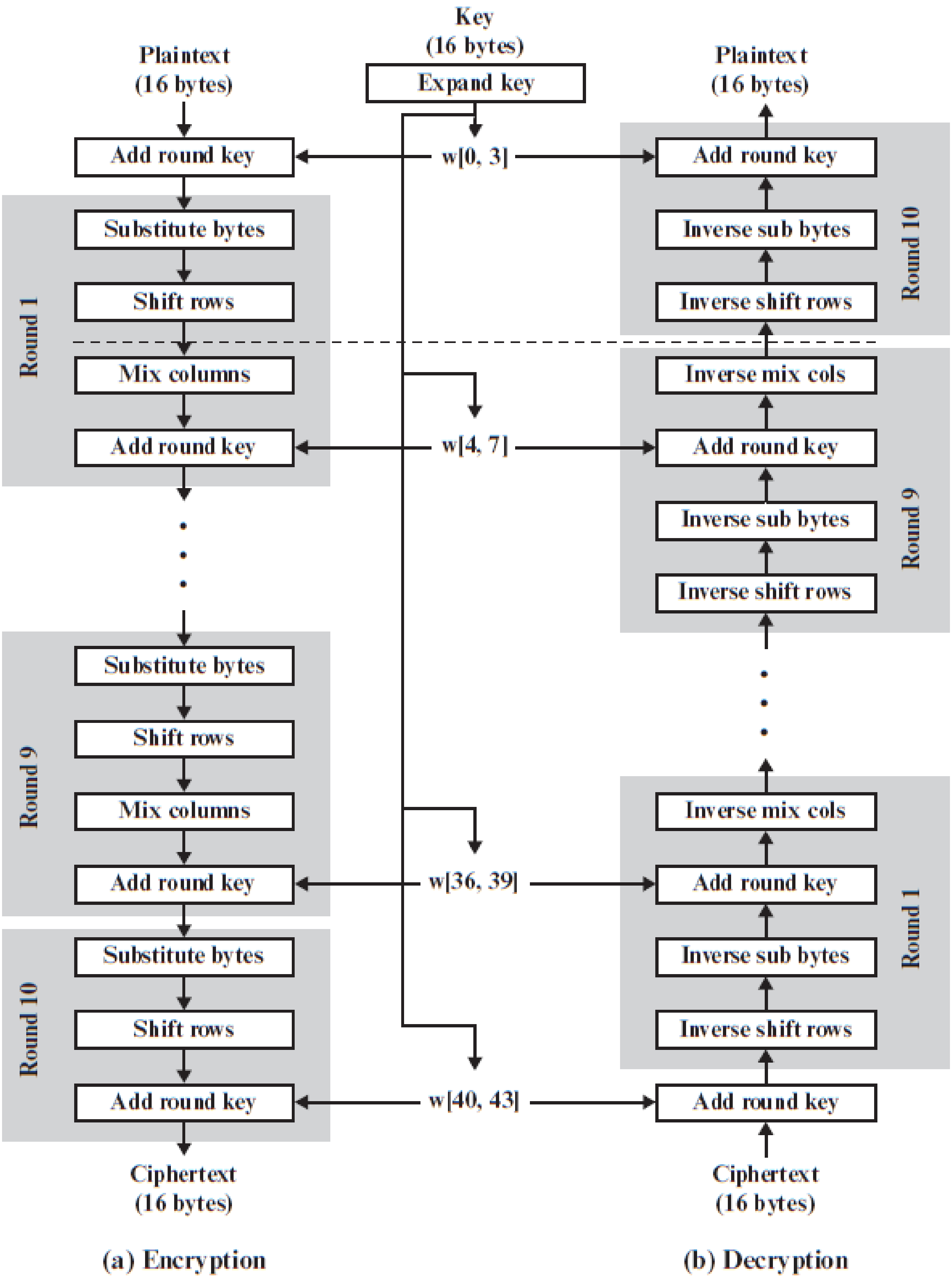


Figure: AES Encryption and Decryption

Applications:

AES is widely used in many applications which require secure data storage and transmission. Some common use cases include:

- **Wireless security:** AES is used in securing wireless networks, such as Wi-Fi networks, to ensure data confidentiality and prevent unauthorized access.
- **Database Encryption:** AES can be applied to encrypt sensitive data stored in databases. This helps protect personal information, financial records, and other confidential data from unauthorized access in case of a data breach.
- **Secure communications:** AES is widely used in protocols like such as internet communications, email, instant messaging, and voice/video calls.It ensures that the data remains confidential.
- **Virtual Private Networks (VPNs):** AES is commonly used in VPN protocols to secure the communication between a user's device and a remote server.

Number Theory:

Number theory, which is the branch of mathematics relating to numbers and the rules governing them, is the mother of modern cryptography – the science of encrypting communication.

Cryptography algorithms that guarantee data privacy, integrity, and authenticity derive their base from number theory, which is utilized to ensure data confidentiality and integrity during online transactions and the identification of digital users.

Applications of Number Theory in Cryptography

Fundamentals of number theory are vital for the formation of modern cryptographic algorithms that assure secure communication and data privacy in many examples of practical use.

Cryptographic algorithms based on the number theory which are already in practice include RSA and elliptic curve among others and are employed in securing online transactions, digital signatures and controlled access systems.

Prime Numbers:

Prime numbers are natural numbers that are divisible by only 1 and the number itself.

In other words, prime numbers are positive integers greater than 1 with exactly two factors,   1 and the number itself.

For example: 2,3,5,7,11

Why Prime Numbers are used in Cryptography?

Prime numbers are used in cryptography due to their role in creating mathematically hard problems, such as integer factorization and discrete logarithms, forming the basis for secure algorithms like RSA and Diffie-Hellman.

Fermat's Little Theorem:

Fermat's theorem states that if p is a prime number, then for any integer a, the number $a^{p} - a$ is an integer multiple of p.

Here p is a prime number

$\boldsymbol{a^p \equiv a \pmod{p}}$**.**

Special Case: If a is not divisible by p, Fermat’s little theorem is equivalent to the statement that a $^{p-1}$-1 is an integer multiple of p.

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

OR

$a^{p-1} \% \ p = 1$

Here a is not divisible by p.

Example :

P = an integer Prime number

a = an integer which is not multiple of P

Let a = 2 and P = 17

According to Fermat's little theorem

$2^{17 - 1} \equiv 1 \text{ mod}(17)$

we got 65536 % 17 ≡ 1

that mean (65536-1) is an multiple of 17

Primality Testing:

Primality test: It is an algorithm for determining whether an input number is prime or not.

Miller-Rabin Primality Test:

Step 1: Find $n-1 = 2^k * m$

Step 2: If $k<=1$;

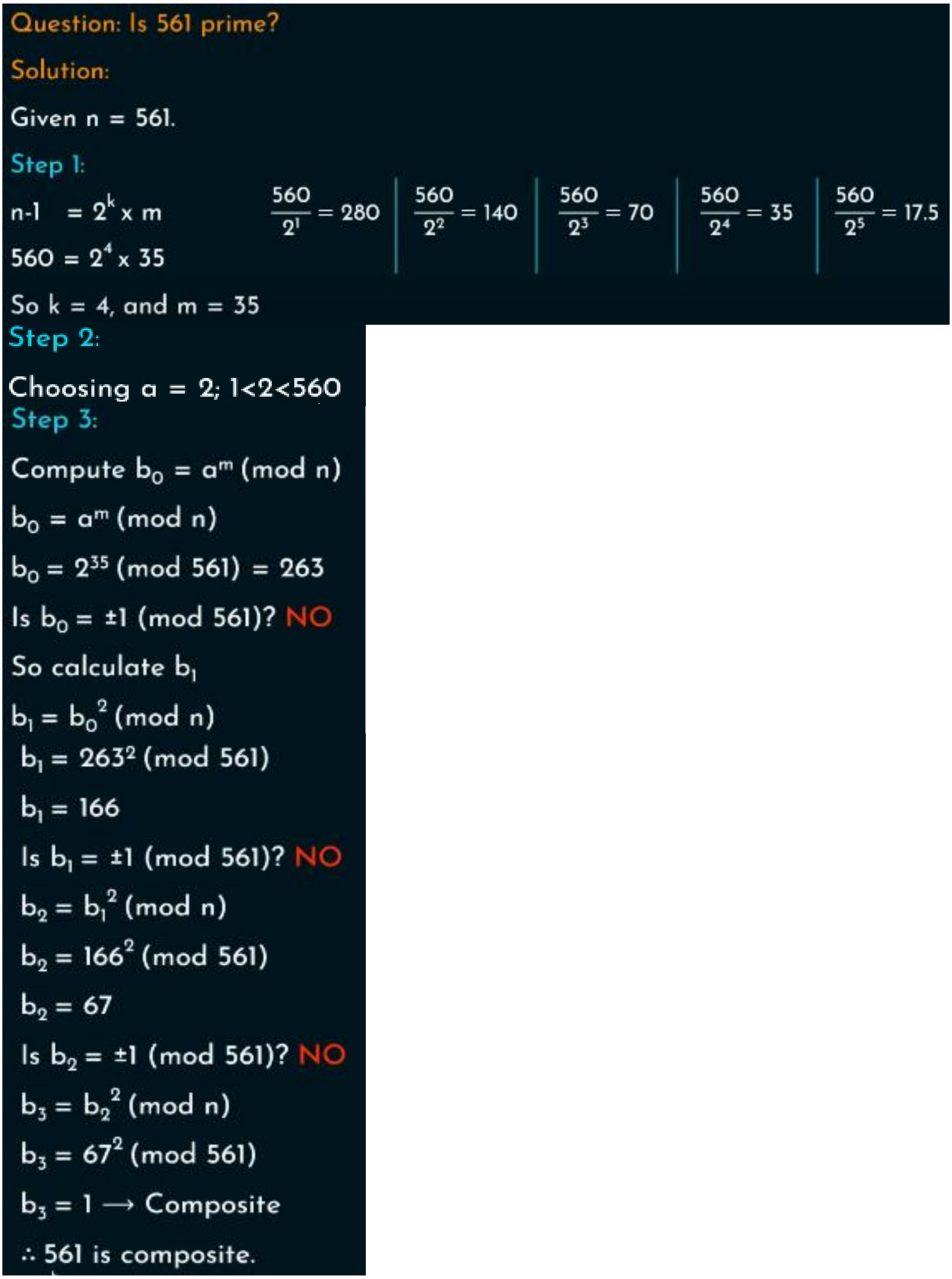
Calculate T such that $T = a^m \bmod n$; if $T = (+-1)$, it is prime, else composite

Step 3: If $k> 1$, Choose ‘a’ such that $1<a<n-1$

Step 4: Compute $b_0=a^m \pmod n$, .., $b_i = b^2_{i-1} \pmod n$

+1 = Composite

-1 = Probably Prime



Question: Is 561 prime?

Solution:

Given n = 561.

Step 1:

$n-1 = 2^k \times m$

$560 = 2^4 \times 35$

So k = 4, and m = 35

$\frac{560}{2^1} = 280$ | $\frac{560}{2^2} = 140$ | $\frac{560}{2^3} = 70$ | $\frac{560}{2^4} = 35$ | $\frac{560}{2^5} = 17.5$

Step 2:

Choosing a = 2; $1<2<560$

Step 3:

Compute $b_0 = a^m \pmod{n}$

$b_0 = a^m \pmod{n}$

$b_0 = 2^{35} \pmod{561} = 263$

Is $b_0 = \pm 1 \pmod{561}$? NO

So calculate b_1

$b_1 = b_0^2 \pmod{n}$

$b_1 = 263^2 \pmod{561}$

$b_1 = 166$

Is $b_1 = \pm 1 \pmod{561}$? NO

$b_2 = b_1^2 \pmod{n}$

$b_2 = 166^2 \pmod{561}$

$b_2 = 67$

Is $b_2 = \pm 1 \pmod{561}$? NO

$b_3 = b_2^2 \pmod{n}$

$b_3 = 67^2 \pmod{561}$

$b_3 = 1 \longrightarrow$ Composite

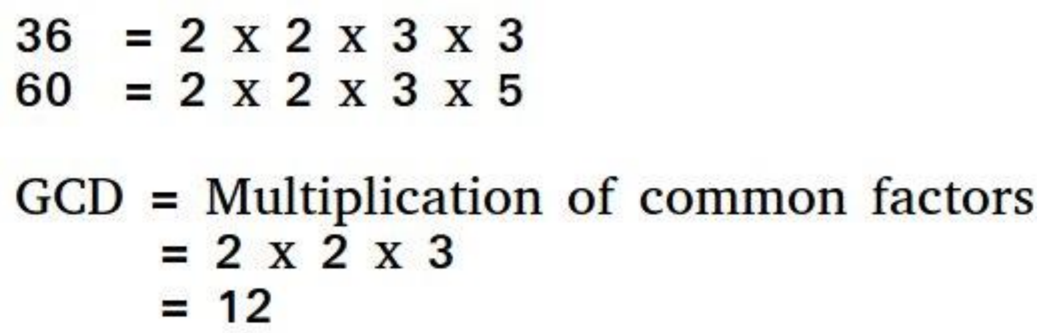
∴ 561 is composite.

Euclidean Algorithm:

The Euclidean algorithm 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.

A simple way to find GCD is to factorize both numbers and multiply common prime factors.



36 = 2 x 2 x 3 x 3

60 = 2 x 2 x 3 x 5

GCD = Multiplication of common factors

= 2 x 2 x 3

= 12

The Euclidean algorithm is a method to find the GCD of two integers a and b. The key ideas behind the algorithm are:

- gcd(a,b)= gcd(b,amodb)
- gcd(a,0)= a

Algorithm Steps

1. Given two positive integers a and b, ensure a≥b
2. If b=0, the GCD is a.
3. Otherwise, replace a with b and b with a mod b

4. Repeat steps 2 and 3 until b becomes 0.
5. The GCD is the last non-zero value of a.

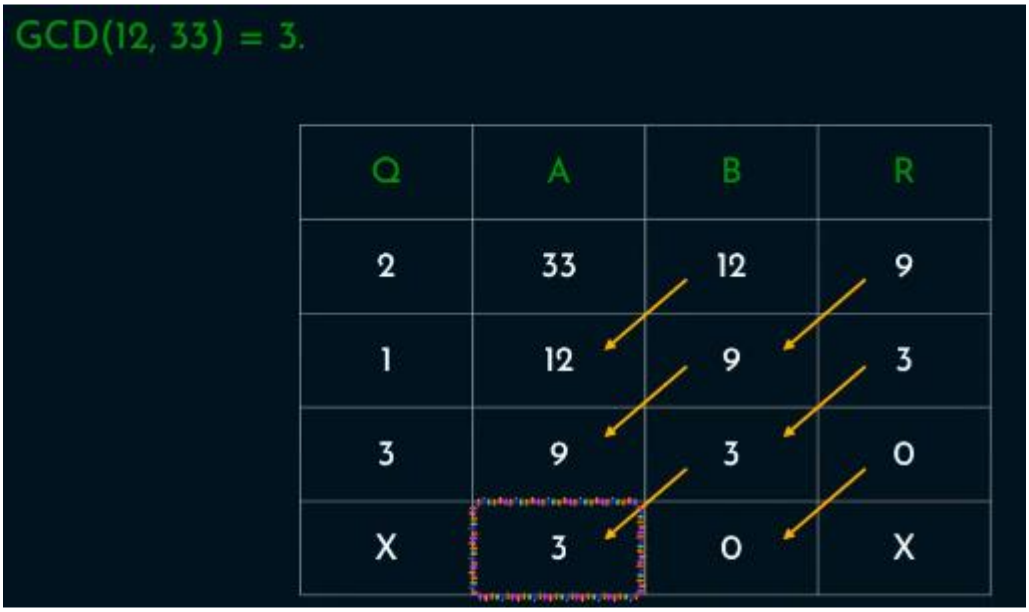
Example

Let's find the GCD of 252 and 105 using the Euclidean algorithm:

1. gcd(252,105):
   - Compute $252 \bmod 105 = 42$
   - Replace a with 105 and b with 42
   - So, gcd(252,105)=gcd(105,42)
2. gcd(105,42):
   - Compute $105 \bmod 42 = 21$
   - Replace a with 42 and b with 21
   - So, gcd(105,42)=gcd(42,21)
3. gcd(42,21):
   - Compute $42 \bmod 21 = 0$
   - Replace a with 21 and b with 0
   - So, gcd(42,21)=gcd(21,0)
4. gcd(21,0)=21

Thus, the GCD of 252 and 105 is 21.

Another Method:



GCD(12, 33) = 3.

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

Extended Euclidean Theorem:

The Extended Euclidean Algorithm is an extension of the Euclidean Algorithm that, besides finding the greatest common divisor (GCD) of two integers a and b, also finds the coefficients s and t of Bézout's identity, which states that:

$$as+bt=GCD(a,b)$$

Q. Find the GCD of (161, 28) and the value of 's' and 't' .

Formula:

$s= s_1-qs_2$

$t= t_1-qt_2$

Solution:

q	a	b	r	S_1	S_2	s	T_1	T_2	t
5	161	28	21	**1**	**0**	1	**0**	**1**	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		**-1**	4		**6**	-23	

So, GCD = 7

S= -1

T= 6

To check answer is correct or not:

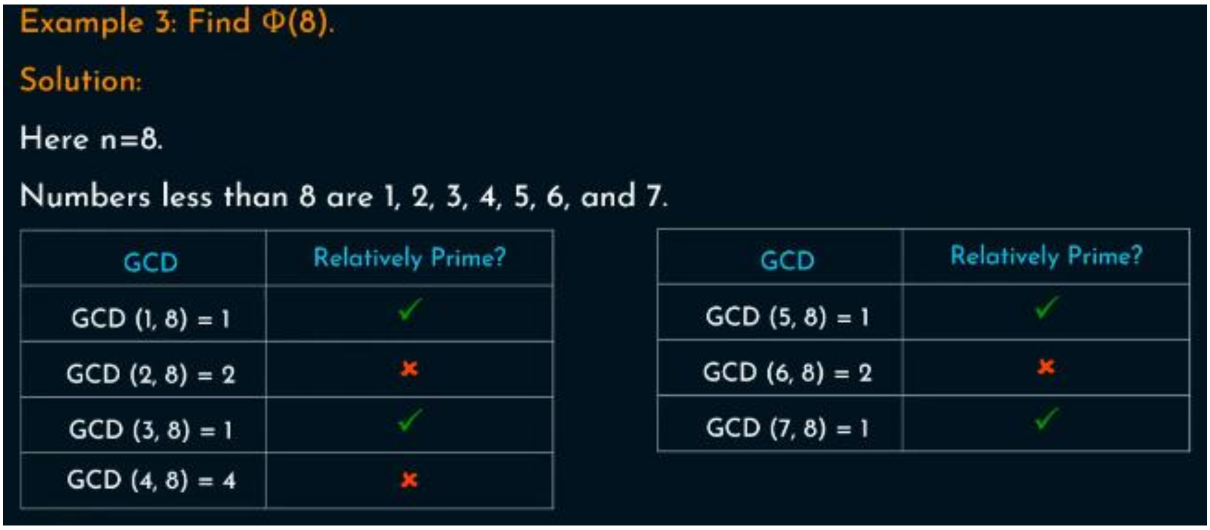
GCD(a,b)= sa+tb =-1*161 + 6*28= 7

Euler’s Totient function:

Relatively prime: When two numbers have no common factors other than 1, they are said to be relatively prime. In other words, no number other than 1 can divide them both exactly (without any remainder)

Euler’s Totient function

- Denoted by Φ(n).
- Φ(n) = Number of positive integers less than ‘n’ that are relatively prime to n



Example 3: Find Φ(8).

Solution:

Here n=8.

Numbers less than 8 are 1, 2, 3, 4, 5, 6, and 7.

GCD	Relatively Prime?
GCD (1, 8) = 1	✓
GCD (2, 8) = 2	✗
GCD (3, 8) = 1	✓
GCD (4, 8) = 4	✗
GCD (5, 8) = 1	✓
GCD (6, 8) = 2	✗
GCD (7, 8) = 1	✓

Φ(8) = 4

Asymmetric Encryption:

Diffie-Helman Key Exchange:

The Diffie-Hellman Key Exchange algorithm is a method for securely exchanging cryptographic keys over a public channel.

This method allows two parties, each with their own private key, to jointly create a shared secret key.

This shared secret can then be used to encrypt subsequent communications.

Algorithm

1. Consider a prime number q
2. Select α such that it must be primitive root of q and α<q

   ‘a’ is a primitive root of q if:

   amod q, a^2 mod q,…, $a^{(q-1)}$ mod q

   gives result:

   $\{1,2,3,\ldots,(q-1)\}$

i.e. Value shouldn't be repeated and we should have all the value in the output set from 1 - (q-1)

Example:

Let q = 7

Calculating primitive root.

Let us assume a= 5 which is less than 7

5^1 mod 7 = 5

5^2 mod 7 = 4

5^3 mod 7 = 6

5^4 mod 7 = 2

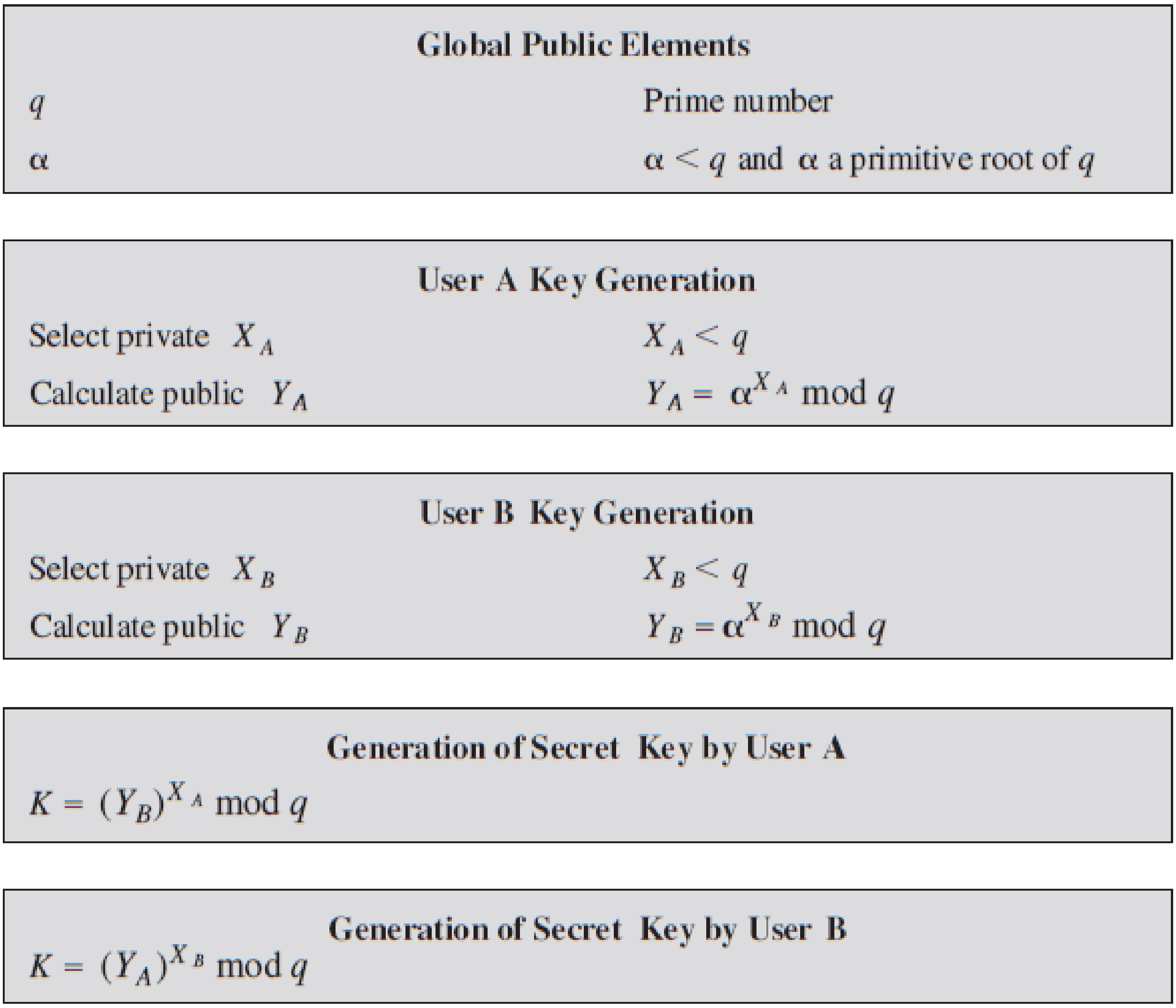
5^5 mod 7 = 3

5^6 mod 7 = 1

‘a’ = 5

Therefore

Here ‘a’ , q is global public elements (known to everyone)



Global Public Elements

q	Prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

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

User B Key Generation

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

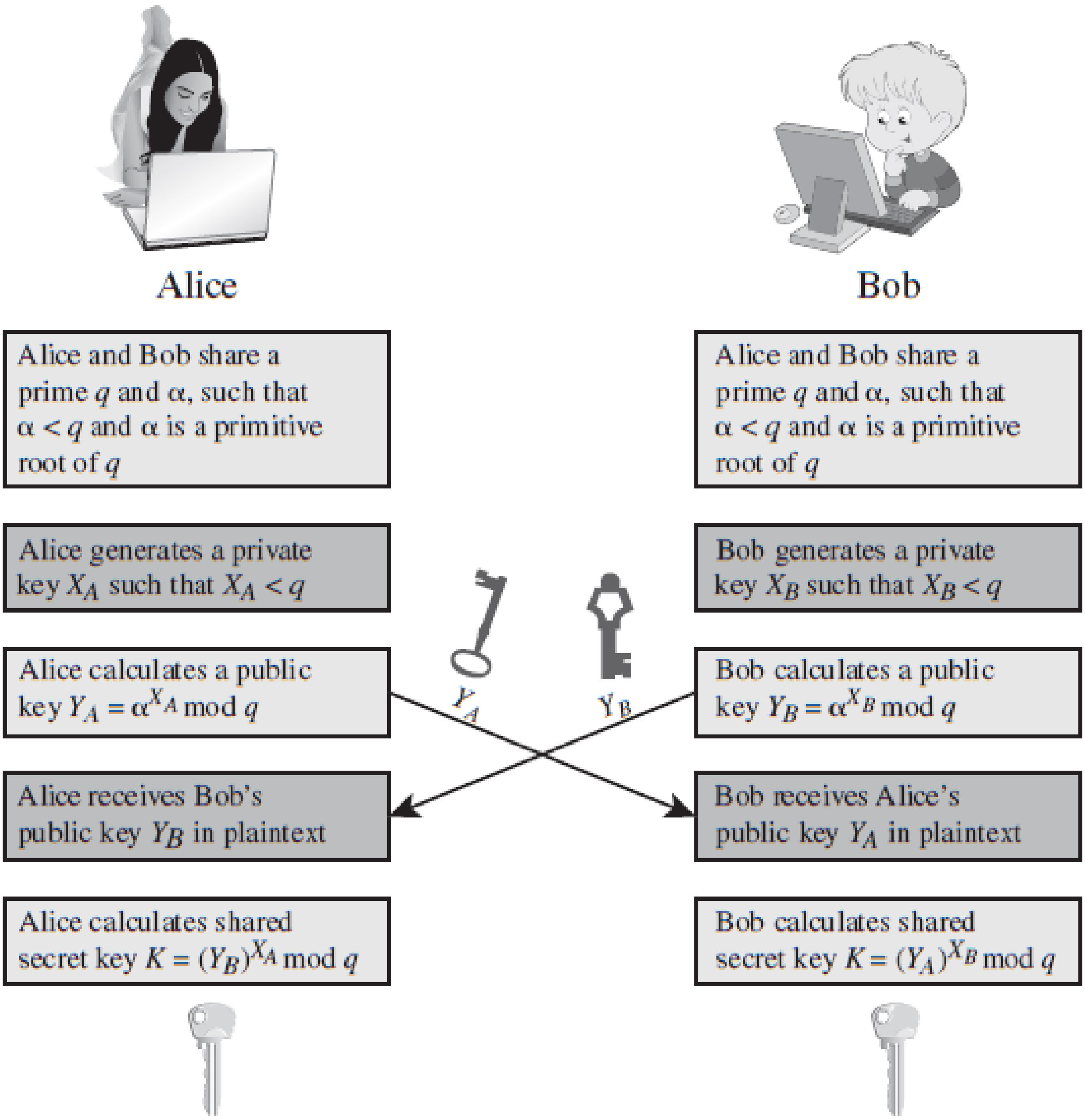
Generation of Secret Key by User A

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

Generation of Secret Key by User B

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

Figure: The Diffie-Hellman Key Exchange Algorithm



Alice and Bob share a prime q and α, such that $\alpha < q$ and α is a primitive root of q

Alice and Bob share a prime q and α, such that $\alpha < q$ and α is a primitive root of q

Alice generates a private key X_A such that $X_A < q$

Bob generates a private key X_B such that $X_B < q$

Alice calculates a public key $Y_A = \alpha^{X_A} \bmod q$

Bob calculates a public key $Y_B = \alpha^{X_B} \bmod q$

Alice receives Bob's public key Y_B in plaintext

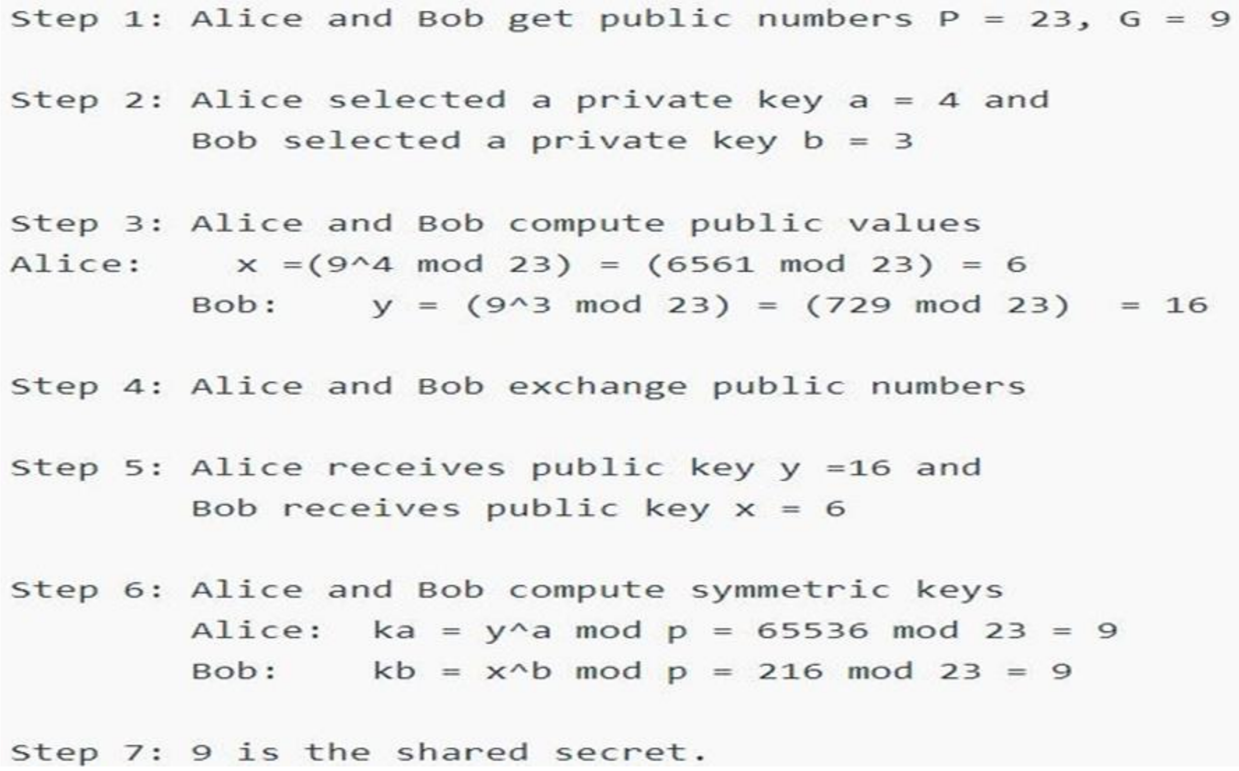
Bob receives Alice's public key Y_A in plaintext

Alice calculates shared secret key $K = (Y_B)^{X_A} \bmod q$

Bob calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

Figure: Diffie-Hellman Key Exchange

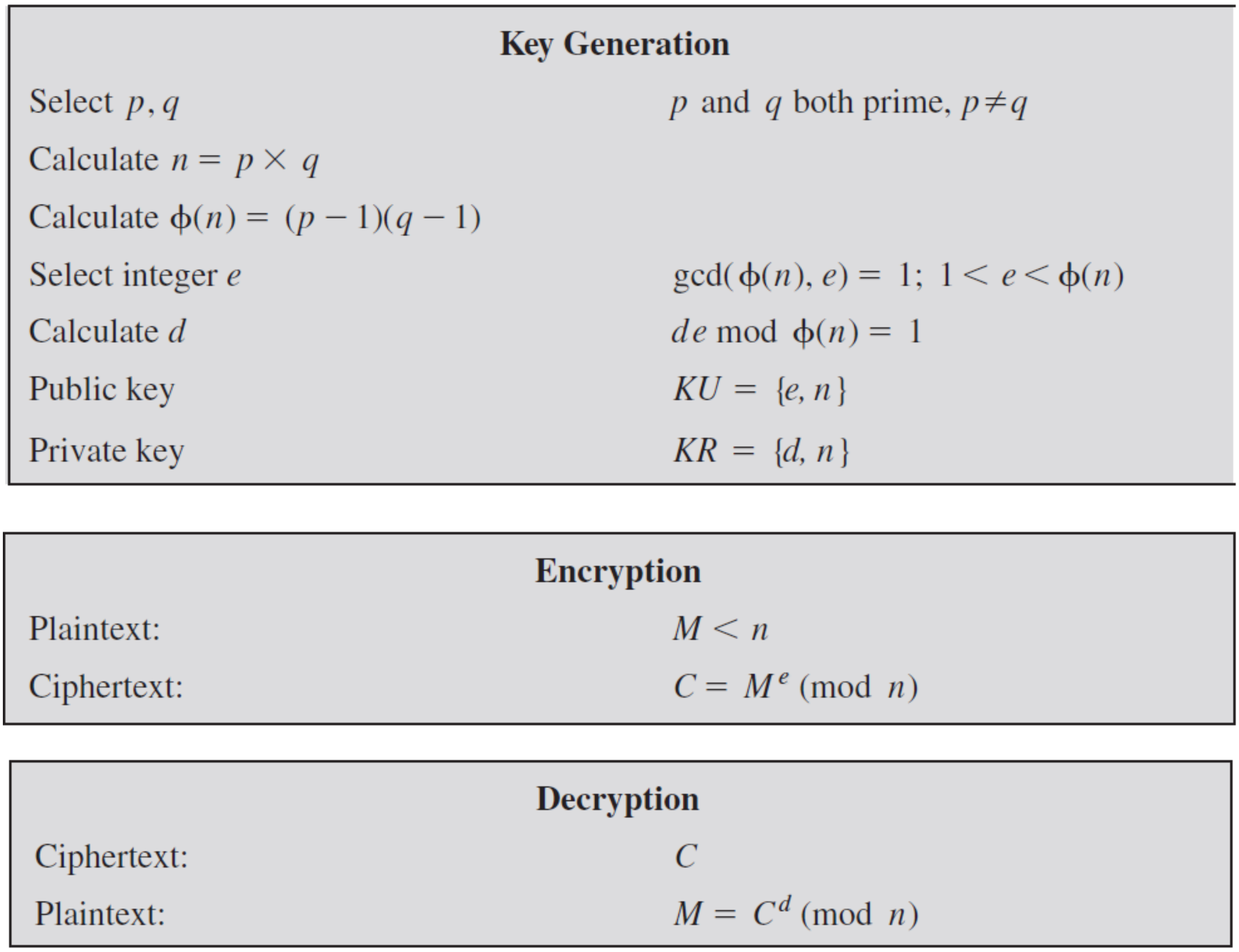
Example:



```
Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
        Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice:    x =(9^4 mod 23) = (6561 mod 23) = 6
        Bob:    y = (9^3 mod 23) = (729 mod 23)  = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
        Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
        Alice:  ka = y^a mod p = 65536 mod 23 = 9
        Bob:    kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.
```

RSA Algorithm:

- It is an asymmetric cryptographic algorithm. I.e. public key and private key
- If public key of user A is used for encryption, we have to use the private key of same user for decryption
- The RSA scheme is a block cipher in which the plain text and cipher text are integers between 0 and n-1 for some value n.
- One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978



Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Figure: The RSA Algorithm

For Example:

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

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

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

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

5. Determine d such that $de \bmod 160 = 1$ and $d < 160$. The correct value is

$$d = 23, \text{ because } 23 * 7 = 161 = (1 * 160) + 1.$$

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

The example shows the use of these keys for a plaintext input of $M = 88$.

For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows:

$88^7 \bmod 187 = [(88^4 \bmod 187) * (88^2 \bmod 187) * (88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$

88^2 mod 187 = 7744 mod 187 = 77

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

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

For decryption, we calculate $M = 11^{23}$ mod 187:

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

11^1 mod 187 = 11

11^2 mod 187 = 121

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

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

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

= 79, 720, 245mod 187 = 88

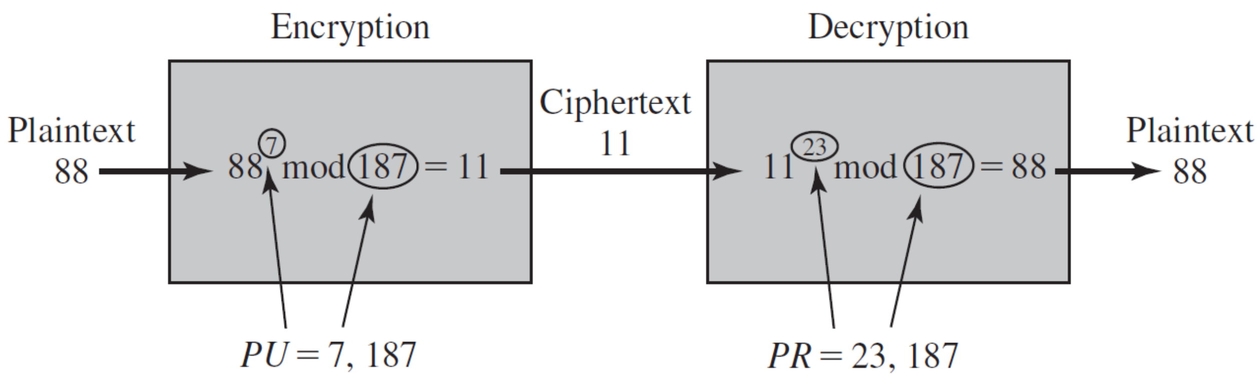


Figure: Example of RSA Algorithm