Crypto

Cryptosystem :

→ set of cryptographic algorithms and protocols design to ensure secure communication and data protection.

→ 3 components :

1. Encryption Algorithm
2. Decryption Algorithm
3. Key management

→ Depending on key, cryptosystem categorized as -

1. Symmetric cryptosystem → same key for both encrypⁿ & decrepⁿ
   faster, less secure — eg: AES, DES
2. Asymmetric cryptosystem → Public key for encryption, Private key for decrepⁿ
   slower, more secure — eg: Elliptic Curve cryptography, RSA, ECC.

→ Key Features :

1. Confidentiality – only authorize party can access
2. Integrity – Detects unauthorized authentication
3. Authentication – verify the identities
4. Non-Repudiation – Ensure sender can't deny sending of msg.

A —m→ B
↑ adversary

instead of m, A sends c (ciphertext)

$C = E(s, m)$

↑ algo. ↑ secret key ↖ message

* Encryption scheme (G, E, D) is a triplet of (possibly probabilistic) algorithms where,
  → key generation $G(1^k)$: outputs secret key (sk) of length k.
  → Encryption algorithm $E(sk, m)$ output cipher text c
  → Decryption algorithm $D(sk, c)$ output plain text m
  $D(sk, E(sk, m))$

Historical Ciphers :

① Shift Cipher (Caesar cipher)

- Encryption algorithm is "shift key characters down" in the alphabet sequence.
- Decryption algorithm is "shift key characters up" in the

n		n	
1	A	14	N
2	B	15	O
3	C	16	P
4	D	17	Q
5	E	18	R
6	F	19	S
7	G	20	T
8	H	21	U
9	I	22	V
10	J	23	W
11	K	24	X
12	L	25	Y
13	M	26	Z

let key = 2 , Plain Text : R I T E S H

Cipher Text : T K V G U J

② Substitution Cipher :

- each element in the plain text is replaced with another element to produce ciphertext.
- The basic idea is to "substitute" each unit of the plain text with a corresponding unit from a different set, according to a fixed system or key.

Types -

(i) Simple substitution cipher

- each letter in the plaintext is replaced by a fixed letter, symbol or number.

eg: Caesar Cipher (Shift Cipher) → monoalphabetic cipher

→ Vigenere Cipher :

- advanced form of subst^n cipher known as polyalphabetic
- uses a series of different caesar ciphers based on repeating the "key".
- hard to break using frequency analysis.

eg: Plain Text : ATTACKATDAWN

key : LEMON

Encryption: ATTACKATDAWN
LEMONLEMONLE

A (shift by 11, since L is the 12th letter) → L

T (shift by 4, since E is the 5th letter) → X

T (shift by 12, since M is the 13th letter) → F

A
:
:

Attacks on Historical Ciphers -

1. Frequency Analysis on Caesar Cipher
2. Kasiski Examination Attack on Vigenere Cipher
3. Cryptoanalytic Bombe on Enigma Cipher
4. Brute Force Attack with Heuristic Technique on Substⁿ Cipher.
5. Linear Attack
6. Differential Attack
7. Integral Attack

---

* Attack Models in Cryptoanalysis -

① Cipher-text-Only Attack (COA):
↳ Attacker has access to the cipher-text (encryted msg). and attempts to deduce the plain text or key.

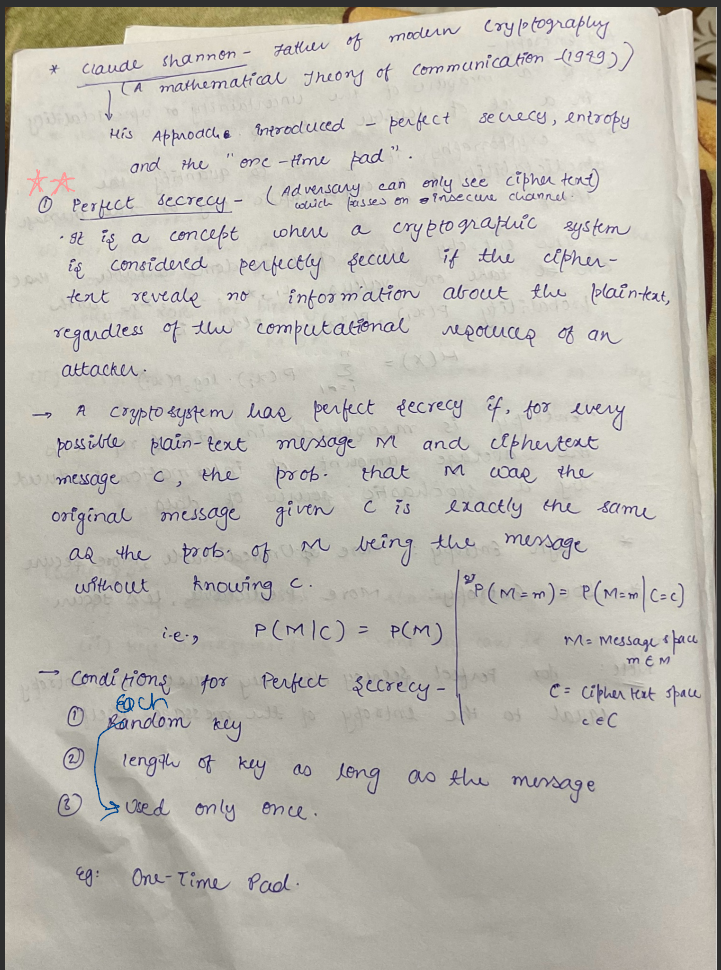
② Known-Plaintext Attack (KPA):
↳ The attacker has access to both plain text and its corr. cipher text.
↳ try to deduce the key or find pattern.

③ Chosen-Plaintext Attack (CPA):
↳ The attacker can choose arbitrary plaintexts and obtain their corr. cipher texts, allowing for more strategic analysis.

④ Chosen-Cipher text Attack (CCA)
↳ The attacker can choose cipher texts and obtain their corr. plaintexts, typically used to find weakness in decryption algorithms.

⑤ Adaptive-Chosen-Plaintext / Cipher text Attack
↳ The attacker can choose plain texts or cipher texts adaptively, based on previous results.
↳ This models reflects a more interactive approach, where the attacker's choices evolve during the cryptanalysis.

→ Shanon's Theorem for Perfect Secrecy :



* Claude Shannon - Father of modern cryptography
(A mathematical Theory of Communication (1949))
↓
His Approach introduced - perfect secrecy, entropy and the "one-time pad".

① Perfect secrecy - (Adversary can only see cipher text which passes on insecure channel)

- It is a concept where a cryptographic system is considered perfectly secure if the cipher-text reveals no information about the plain-text, regardless of the computational resources of an attacker.

→ A cryptosystem has perfect secrecy if, for every possible plain-text message M and ciphertext message C, the prob. that M was the original message given C is exactly the same as the prob. of M being the message without knowing C.

i.e., $P(M|C) = P(M)$

$\forall P(M=m) = P(M=m|C=c)$

M = Message space $m \in M$

C = Cipher text space $c \in C$

→ Conditions for Perfect Secrecy -

① Each Random key

② length of key as long as the message

③ Used only once.

Eg: One-Time Pad.

Statement of Shanon's Theorem :

- A cryptosystem achieve perfect secrecy if -

$$I(M;C) = 0$$

where $I(M;C)$ is the mutual information b/w the plain-text message M & cipher text C

- The probability of a plain-text given a ciphertext and a key must be the same as the probability of the plaintext without knowing the ciphertext

⇒ The ciphertext does not reduce the uncertainity about the plaintext

Proof :

- Let M be the Plaintext / Message
  K be the key
  C be the ciphertext
- For perfect secrecy, the prob. distrⁿ of M must remain unchanged regardless of the cipher text C.

$$P(M|C) = P(M)$$

Conditional Property of Plain-text

Unconditional Property of Plaintext

→ This can only happen if "each key 'K' is used only once" (i.e, a one-time pad) and "length of the key is atleast as long as the message".

Kirchoff's Principle :

- The security of a cryptosystem should rely solely on the "secrecy of key" and not on the secrecy of the algorithm.

→ Implications of Kirchoff's law :

① Public Algorithms : assuming they are Publicly known.

② Key security : security of system depend on key.

③ Transparency : independently verified system

③ One-Time Pad - (OTP)

- OTP is a cryptographic technique that shannon proved to achieve perfect secrecy.
- It is a symm. encryption method that uses a random key that is as long as the message itself.

→ How it works -

(i) Encryption : Each bit of plaintext is combined with the corresponding bit of the key using an oper^n (like bitwise XOR for binary data).

$$C = M \oplus K$$

(ii) Decryption : To decrypt the ciphertext, the same key is used to reverse the encryption process.

$$M = C \oplus K$$

→ Now Cypher text is statistically independent of the plain text.

Challenges -

(i) key distribution : sharing of key is insecure

(ii) key management : one key can be used only once

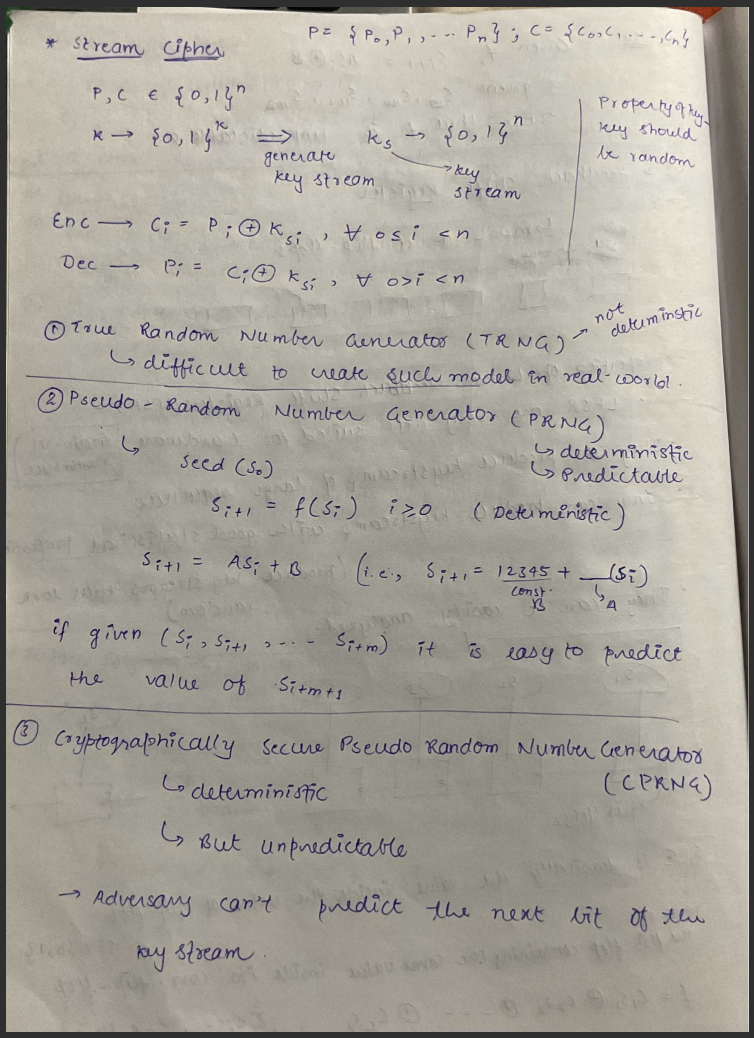
A **stream cipher** is a type of symmetric key cipher in which plaintext digits are combined with a pseudorandom cipher digit stream (the keystream). Rather than encrypting data in fixed-size blocks (as in block ciphers like DES or AES), stream ciphers encrypt plaintext one bit or byte at a time.

Key Characteristics of Stream Ciphers:

- **Real-time Encryption**: Stream ciphers are well-suited for real-time encryption where data is transmitted as a continuous stream, such as in communication protocols.
- **Keystream Generation**: Stream ciphers generate a keystream from the encryption key using an internal state and combine this keystream with the plaintext (typically using XOR) to produce the ciphertext. Decryption involves applying the same keystream and XOR operation to the ciphertext.
- **Speed and Efficiency**: Stream ciphers are generally faster and more efficient than block ciphers for certain applications, especially when the data is not available in fixed blocks.
- **Key Synchronization**: Both the sender and receiver must maintain synchronized states to ensure proper encryption and decryption.

Examples of Stream Ciphers:

1. **RC4**: One of the most well-known stream ciphers, although it is now considered insecure for many uses due to vulnerabilities in its keystream.
2. **ChaCha**: A modern stream cipher known for its security and efficiency, widely used in protocols like TLS (Transport Layer Security).
3. **Salsa20**: Another secure stream cipher designed for high-performance encryption.



Aspect	TRNG	PRNG	CSPRNG
Source of Randomness	Physical phenomena	Deterministic algorithm	Deterministic algorithm
Reproducibility	Not reproducible	Reproducible with the same seed	Reproducible but focused on unpredictability
Security	Highly random but not always practical for cryptographic use	Not secure, predictable if seed is known	Secure and unpredictable, even if part of the output is known
Use Case	Cryptographic key generation, hardware-based randomness	Simulations, games, non-security applications	Cryptography, secure communications
Speed	Slower due to physical processes	Fast and efficient	Fast but more secure than PRNG

TRNGs provide true randomness from physical phenomena, PRNGs use algorithms for reproducible pseudorandomness, and CSPRNGs are designed for security in cryptographic systems.

Ans

Given f, $S_{i+1} = AS_i \oplus B$

given $S_{99}, S_{100}, S_{101}, S_{102}$

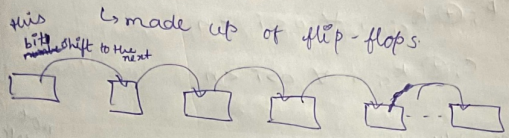
Show that f is unpredictable ?

→ Q01: XOR with a const B introduces Non-linearity, since the values of A & B is unknown.

Also, Invertible, since recovering S_i from S_{i+1} without knowing A & B is very difficult

→ This adds Unpredictability to the system, especially since multiple states like $S_{99}, S_{100}, S_{101}, S_{102}$ are provided without clear knowledge of the underlying structure.

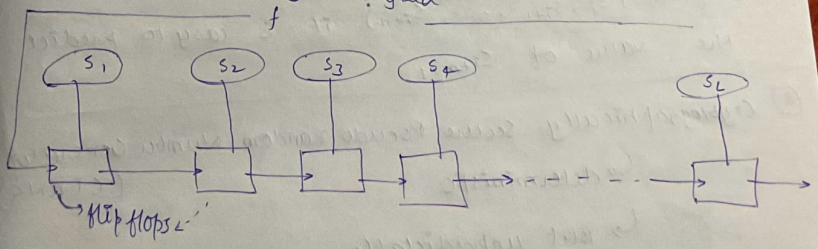
* Feedback Shift Register



↳ made up of flip-flops

→ Properties of Linear feedback shift Registers (LFSR) –

- LFSR are very well suited for hardware (Grain-V1 Martin Hell)
- They can produce keystreams of large sequences.
- They can produce keystreams with good statistical properties (produce key streams that look random)
- They can be easily analyzed.



S_1, S_2 is basically the value inside the value of flip-flop

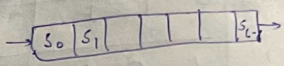
↓ name of the flip-flop containing the same value inside its corr. flip-flop

$s_i \in \{0,1\}$

$f = C_1S_1 \oplus C_2S_2 \oplus \cdots \oplus C_LS_L$, $\sum_{i=1}^{L} C_iS_i$, $C_i \in \{0,1\}$

* LFSR - definition -

- A LFSR of length L consists of L flip-flops numbered $0, 1, \ldots L-1$, each capable of storing 1 bit (0/1), and a clock which controls the movement of data, then at each clocking -



① The content of S_{L-1} will be the output and form part of output sequence.

② The contents of S_i will move to S_{i+1}

i.e.,
$$S_{i+1} = S_i \quad ; \quad 0 \le S_i \le L-2$$

③ The new value of S_0 will be the feedback bit f_0 defined as the XOR of the values in a fixed subset of $\{S_0, S_1, \ldots, S_{L-1}\}$ of the previous clock.

④ At clock 0, the LFSR is initialised by a seed and this stage is called "initial stage".

---

* Feedback Polynomial / Connecting Polynomial - Defn -

- An LFSR $(L, C(S))$, where $C(S)$ is defined as

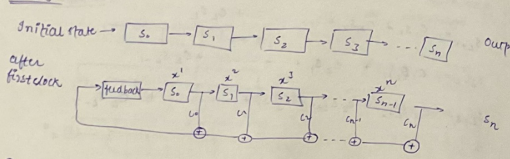
~~$C(S) = 1 + c_0 s_0 + c_1 s_1 + \ldots + c_{L-1} s_{L-1}$~~ ; where $c \in \{0,1\}$, $s \in \{0,1\}$

(L = length; c_0 = 1)

and is known as feedback / connecting polynomial.

$$C(S) = 1 + c_1 S^1 + c_2 S^2 + c_3 S^3 + \ldots + c_L S^L$$

LFSR



① $(L, C(S)) \longrightarrow L, C(S) = 1 + c_1 S + c_2 S^2 + \cdots + c_L x^L$

② $L, s_{t+1} = c_1 s_1 + c_2 s_2 + c_3 s_3 + \cdots + c_L s_L$

③ $L, P(x) = 1 + c_1 x + c_2 x_2 + \cdots c_L x_L$

- Theorem → The max^m length of the sequence generated by a LFSR $(L, C(S))$ will be $2^L - 1$.

---

* Known plaintext Attack on A single LSFSR :

2m : known plaintext bits ⇒ 2m keystream bits
(also, by default the adversary knowns the ciphertext bits)

$$s_{i+L} = \sum_{j=1}^{L} c_j s_{i+j-1}, \quad i = 1, 2, \cdots, \quad j = 1, 2, \cdots L$$
$$c_j, s_j \in \{0, 1\}$$

$i=0$ $\boxed{s_1 \mid \cdots \mid s_L}$

$i=1$ $\boxed{s_{L+1} \mid s_1 \mid \cdots \mid s_{L-1}}$ $\longrightarrow O_1 = s_L$

$i=2$ $\boxed{s_{L+2} \mid s_{L+1} \mid s_1 \mid \cdots \mid s_{L-2}}$ $\longrightarrow O_2 = s_{L-1}$

$i=3$ $\boxed{s_{L+3} \mid s_{L+2} \mid s_{L+1} \mid s_1 \mid \cdots \mid s_{L-3}}$ $\longrightarrow O_3 = s_{L-2}$

Example

This 1 will not be while doing XOR

LFSR $(3, 1+s+s^3)$

initial state → 0 1 1

Initial	⊕		output bit (shown bit)	
	0	1	1	
1	1	0	1	
2	0	1	0	
3	0	0	1	
4	1	0	0	
5	1	1	0	
6	1	1	1	
7	0	1	1	
8				

LSFR $(3, 1+s+s^3) \rightarrow s_{t+3} = s_{t+2} + s_t$

LSFR $(3, 1+s+s^3) \rightarrow$ nearly upto $2^3 - 1$

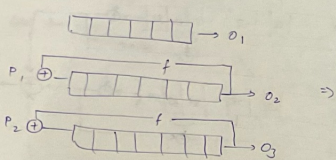
∴ it is creating pattern by always given the last element from the stream. XOR

⇒ so, we will output the LFSR's output.

→ Periodicity of LFSR

* Let $C(S)$ be a connecting polynomial

Asynchr. used for pre-known plaintext or message.

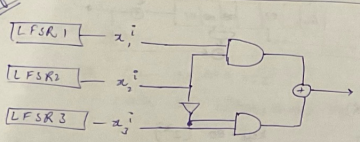


$P_i + O_i = C_i$

$P_1 + O_1 = C_1$

$P_2 + O_2 = C_2$

Cuffe



$f = x_1^i \cdot x_2^i \oplus (1 + x_2^i)\, x_3^i$

$f = x_1^i x_2^i \oplus x_2^i x_3^i \oplus x_3^i$

x_1^i	x_2^i	x_3^i	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$s_{L+1} = c_1 s_1 + c_2 s_2 + \cdots + c_L s_L$

$s_{L+2} = c_1 s_{L+1} + c_2 s_1 + \cdots + c_L s_{L-1}$

$s_{L+3} = c_1 s_{L+2} + c_2 s_{L+1} + c_3 s_1 + \cdots + c_L s_{L-2}$

$\vdots$

$s_{2L-1} = c_1 s_{2L-2} + c_2 s_{2L-3} + \cdots + c_L s_{2L-L-2}$

$s_{2L} = c_1 s_{2L-1} + c_2 s_{2L-2} + \cdots + c_L s_{2L-L-1}$

NFSR (only difference the feedback function is Non-Linear)

f_1 : $s_1 \oplus s_3$ (Linear only using XOR)

f_2 : $s_1 \oplus s_2 \cdot s_3$ (Non linear also using AND)

---

① Synchronous stream ciphers

② Asynchronous

Basically the key streams (KSi) ← $O_i = f(L, k)$

$C_i = P_i + O_i$ } Synchr.

$O_1 = f(L, k)$

$O_2 = f(L, k, P_i^*)$

* Components of Encryption-algo
  1. circular shift
  2. Permutation
  3. Substitution → linear / non-linear (bijective $n \to n$, NI $n \to m$; $n > m$)
  4. Exclusive-OR
  5. Swapping

→ Components is invertible ✓
→ Components is non-invertible ??

* Block ciphers can be broadly categorized into
  1. Feistel ciphers (invertible, non-invertible components (DES))
  2. Non-feistel ciphers. (AES)
     ↓
     These class of ciphers uses only invertible components in the units of Encryption algorithm.

** In the non-invertible, at the end we have to use XOR. to cancel out the non-invertibility component.

Imp Ques: How does Feistel ciphers makes use of non-invertible components in its encryption algorithm (i.e., the property of which component can be used to cancel out the non-invertibility.

Eg: Explain with an example

* Confusion and Diffusion

Diffusion - The property hides the rel^n b/w cipher text and plain text.

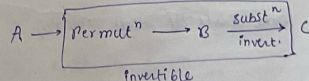
Confusion - The property hides the rel^n b/w cipher text and key.

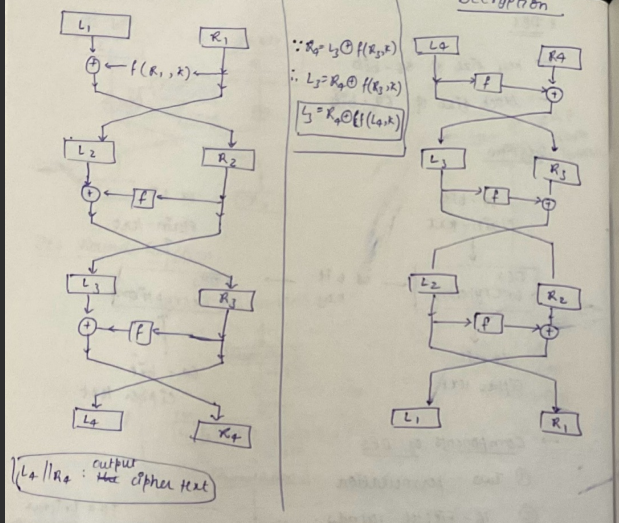
changing one bit of plain text should affect the almost all the cipher text
↓
Avalanche effect.

Completeness effect -
All most all the cipher text bits should depend on every bit of plain text & key.

**

"The units used in an encryption algo. should be invertible".



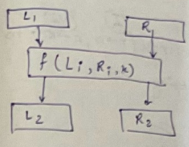


$\because R_2 = L_3 \oplus f(R_3, k)$
$\therefore L_3 = R_2 \oplus f(R_3, k)$
$L_3 = R_4 \oplus f(L_4, k)$

⇒ The effects of the non-invertible components in the Encryption algorithm can be cancelled out in the decryption algorithm by the use of XOR operator.

→ How does a Feistel structure look like

* Feister structure:

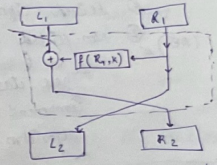


General structure
Plain text
↓ encryption round (K)
Cipher text

⇒ Modified Feistel structure

$L_2 \leftarrow R_1$

$R_2 \leftarrow f(R_1, k) \oplus L_1$



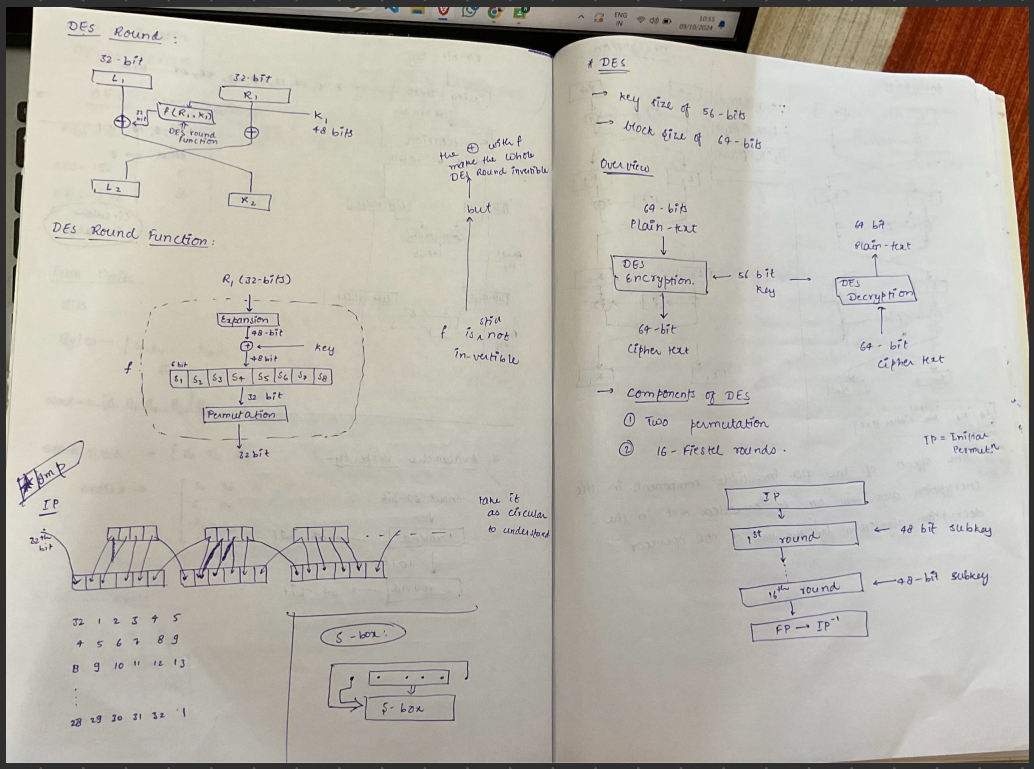
• Let $P = L_1 \| R_1$ be the plain-text divided equally into two halves L_1 and R_1; the block $(L_2 \| R_2)$ after 1 round can be computed as

$L_2 \leftarrow R_1$

$R_2 \leftarrow L_1 \oplus f(R_1, k)$

concatenation

whole unit becomes invertible due to the XOR.



Summary of Differences:

Feature	Synchronous Stream Cipher	Asynchronous Stream Cipher
Keystream Generation	Independent of plaintext or ciphertext, based on key and IV	Depends on previous ciphertext and key
Synchronization Requirement	Must remain perfectly synchronized	Automatically resynchronizes after a few ciphertext blocks
Error Propagation	No error propagation beyond the lost/corrupted ciphertext	Temporary error propagation, self-recovering after a few bits
Complexity	Simpler, as keystream is generated independently	More complex, since the keystream depends on previous ciphertext
Examples	RC4, A5/1 (used in GSM encryption)	Cipher Feedback Mode (CFB), Output Feedback Mode (OFB)
Use Cases	Real-time streaming, video/audio encryption	Wireless communication, unreliable channels

Properties of DES –

① Symm. key cipher
② Block cipher
③ Feistel Network structure
④ IP & FP
⑤ 16 Rounds encryption
⑥ S-Boxes and P-Boxes

Feistel Cipher

A **Feistel cipher** is a symmetric structure used in many block cipher designs. It involves dividing the plaintext into two halves, processing one half using a function (known as the Feistel function), and then combining the result with the other half. The key characteristic is that encryption and decryption processes are nearly identical, making Feistel ciphers efficient for implementation. Some key features:

- Each round of processing applies a function to one half of the data and then swaps the halves.
- It relies on multiple rounds of encryption to increase security.
- The original Feistel network was introduced by Horst Feistel and was used in the **DES** algorithm.

Data Encryption Standard (DES)

DES (Data Encryption Standard) is a symmetric-key block cipher based on the Feistel structure. It was adopted as a standard in 1977 by NIST (then called the National Bureau of Standards). DES operates on 64-bit blocks and uses a 56-bit key, providing moderate security in the early years of computing. Key characteristics:

- DES divides the data into two halves and applies 16 rounds of Feistel transformations.
- Each round involves permutation, expansion, substitution using S-boxes, and XOR operations with round keys.
- Due to its relatively short key size (56 bits), DES is vulnerable to brute-force attacks today, but it was widely used before being replaced by more secure algorithms like AES.

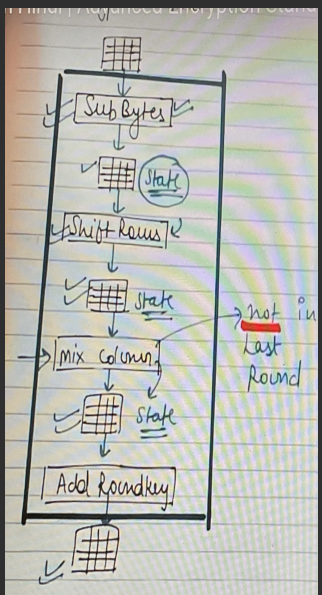
Advanced Encryption Standard (AES)

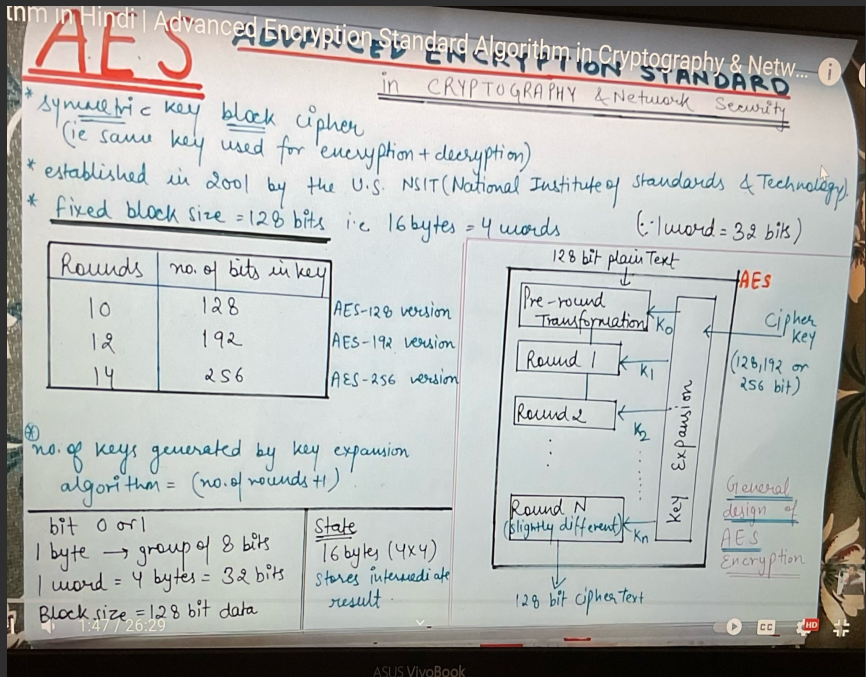
AES (Advanced Encryption Standard) is a symmetric-key block cipher that replaced DES due to its stronger security. AES was selected as the encryption standard by NIST in 2001 after a global competition. Unlike DES, AES does not use a Feistel structure. Instead, it uses a substitution-permutation network. Key features:

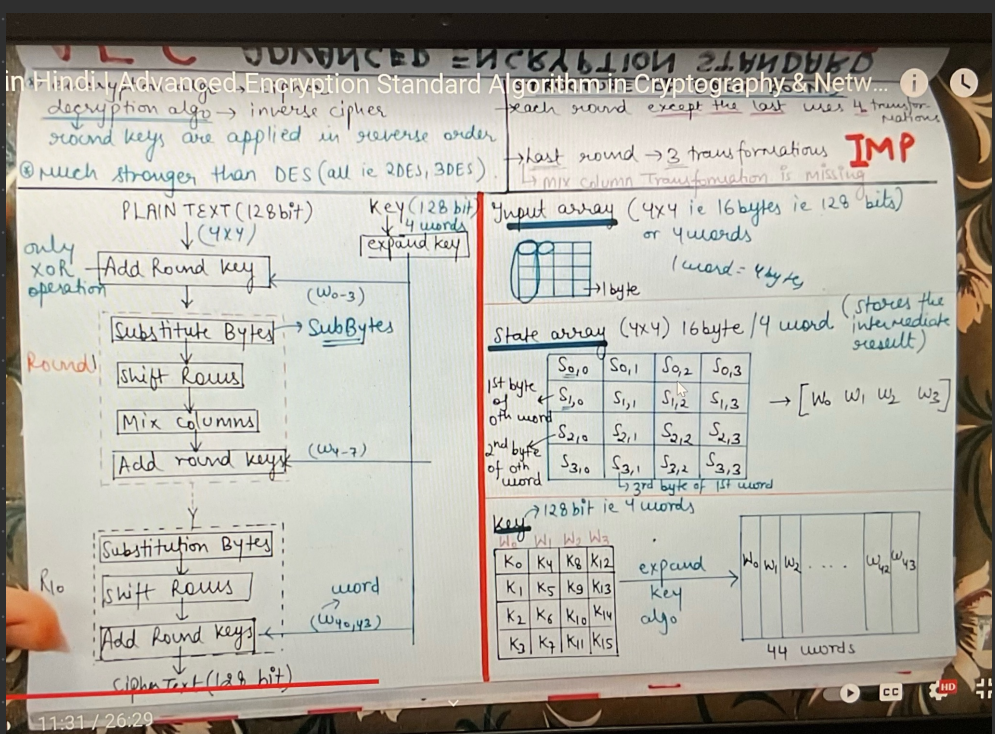
- AES operates on 128-bit blocks of data.
- It supports key sizes of 128, 192, or 256 bits.
- It involves multiple transformations: **SubBytes** (byte substitution), **ShiftRows** (row shifting), **MixColumns** (mixing columns using matrix multiplication), and **AddRoundKey** (XOR with a round key).
- AES is widely used today for secure communication due to its resistance to most forms of attack, including brute-force attacks.

In summary, Feistel ciphers like DES rely on splitting data into halves, while AES uses a more modern approach with byte-level operations for stronger security.

AES


Sub Bytes
State
Shift Rows
State
Not in Last Round
Mix Columns
State
Add Roundkey


AES
ADVANCED ENCRYPTION STANDARD
in CRYPTOGRAPHY & Network Security
* symmetric key block cipher (ie same key used for encryption + decryption)
* established in 2001 by the U.S. NSIT (National Institute of Standards & Technology)
* fixed block size = 128 bits i.e. 16 bytes = 4 words (∵ 1 word = 32 bits)
Rounds | no. of bits in key
10 | 128 | AES-128 version
12 | 192 | AES-192 version
14 | 256 | AES-256 version
* no. of keys generated by key expansion algorithm = (no. of rounds + 1)
bit 0 or 1
1 byte → group of 8 bits
1 word = 4 bytes = 32 bits
Block size = 128 bit data
State
16 bytes (4x4)
stores intermediate result
128 bit plain Text
Pre-round Transformation
Round 1
Round 2
Round N (slightly different)
Key Expansion
Cipher Key (128, 192 or 256 bit)
General design of AES Encryption
128 bit cipher text
1:47 / 26:29


decryption algo → inverse cipher
round keys are applied in reverse order
much stronger than DES (all ie 2DES, 3DES)
each round except the last uses 4 transformations
last round → 3 transformations
mix column Transformation is missing
IMP
PLAIN TEXT (128 bit)
(4x4)
Key (128 bit)
4 words
expand key
only XOR operation
Add Round key
(W0-3)
Substitute Bytes
SubBytes
Round 1
Shift Rows
Mix Columns
Add round key
(W4-7)
Substitution Bytes
Shift Rows
Add Round Keys
Round 10
word (W40,43)
Cipher Text (128 bit)
Input array (4x4 ie 16 bytes ie 128 bits) or 4 words
1 word = 4 bytes
1 byte
State array (4x4) 16 byte / 4 word (stores the intermediate result)
1st byte of 0th word
2nd byte of 0th word
3rd byte of 1st word
[W0 W1 W2 W3]
Key → 128 bit ie 4 words
expand key algo
44 words
11:31 / 26:29

TRANSFORMATIONS → 4 types: substitution, permutation, mixing, and key-adding

1) Substitution

AES, like DES uses substitution. But mechanism is dif. Substitution is done for each byte

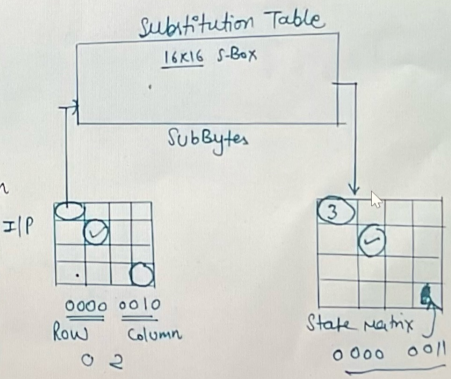
Only one table is used for transformation of bytes, which means that if 2 bytes are same, the transformation is also same.

SubBytes - at encryption side

We interpret the byte as 2 hexadecimal digits.

1st hexadecimal digit → row
2nd hexadecimal digit → column
} of the substitution Table

Transformation is done one byte at a time.



2. Permutation → In this we perform operations at bit level.

In DES, permutation was done at bit level.
AES, " " is " " byte level.

ShiftRows

* Shifting is done to the Left
* no. of shifts depends on the row of the state matrix.

Row 0	63	C9	FE	30
Row 1	F2	F2	63	26
Row 2	C9	C3	7D	D4
Row 3	BA	63	82	D4

Shift Rows (shift left) →

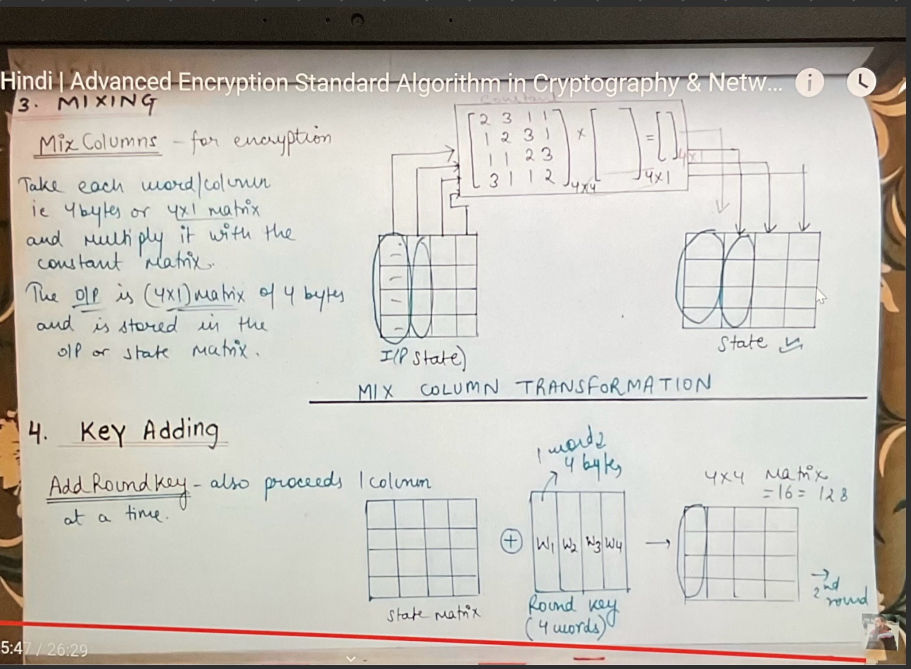
63	C9	FE	30
F2	63	26	F2
7D	D4	C9	C3
D4	BA	63	82

16 bytes

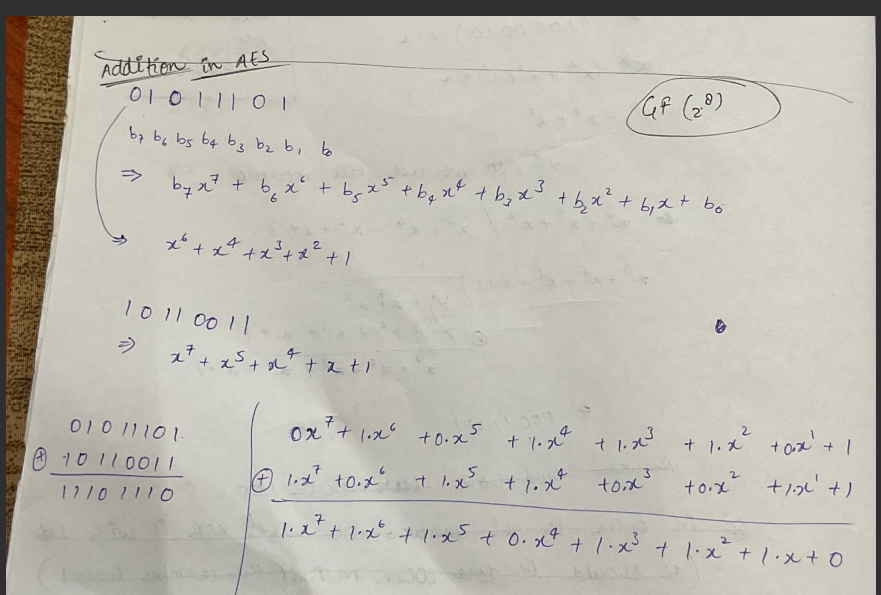
0 or No shift ✓
1-byte shift
2 byte shift
3 byte shift

In decryption, we use InvShiftRows (shifting is to the right)
no. of shifts → same.

Note → ShiftRows & InvShiftRows } transformations are inverses of each other.



Addition in AES :



02 = 0000 0010 = x

01 = 0000 0001 = 1

03 = 0000 0011 = $x+1$

Modulo polynomial is : $x^8+x^4+x^3+x+1$

Eg: 0100 0010 = (x^6+x)

$(0100\ 0010) * (x)$

$(x^6+x)*(x) = x^7+x^2$

$= 1000\ 0100$

Note: it is getting shifted to left 1 time

Eg: 1100 0010 = (x^7+x^6+x)

$(1100\ 0010) * x$

$(x^7+x^6+x)*x$

$x^8+x^7+x^2$

$\because x^8$ so we use the modulo

$\therefore$ & $x^8+x^7+x^2 \,/\, x^8+x^4+x^3+x+1$

$$x^8+x^4+x^3+x+1 \overline{)\ x^8+x^7+x^2\ (\ 1}$$

$$\pm\ x^8+x^4+x^3+x+1$$

$$x^7+x^4+x^3+x^2+x+1$$

Answer would be 10000100

$= 1001\ 1111$

To get the answer by doing left shift

i.e., 1100 0010

← left shift 1 time (if multiply with 02 i.e. equal to x)

= 1000 0100

Now XOR it with 1B i.e. equal to 0001 1011

$\Rightarrow$

$$\begin{array}{r} 1000\ 0100 \\ 0001\ 1011 \\ \hline 1001\ 1111 \end{array}$$

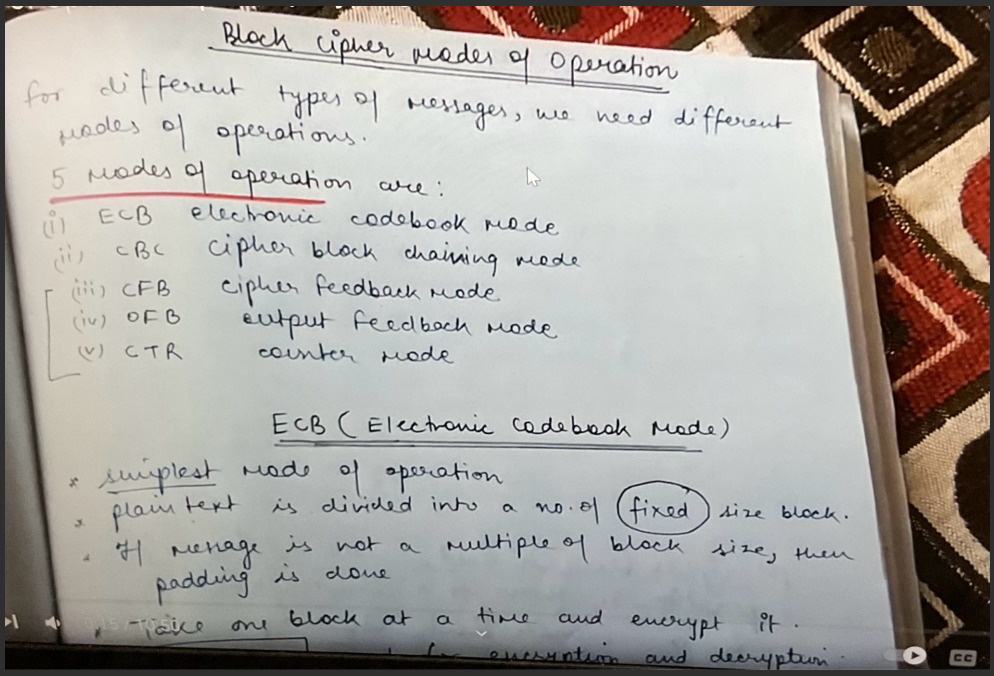
→ correct answer

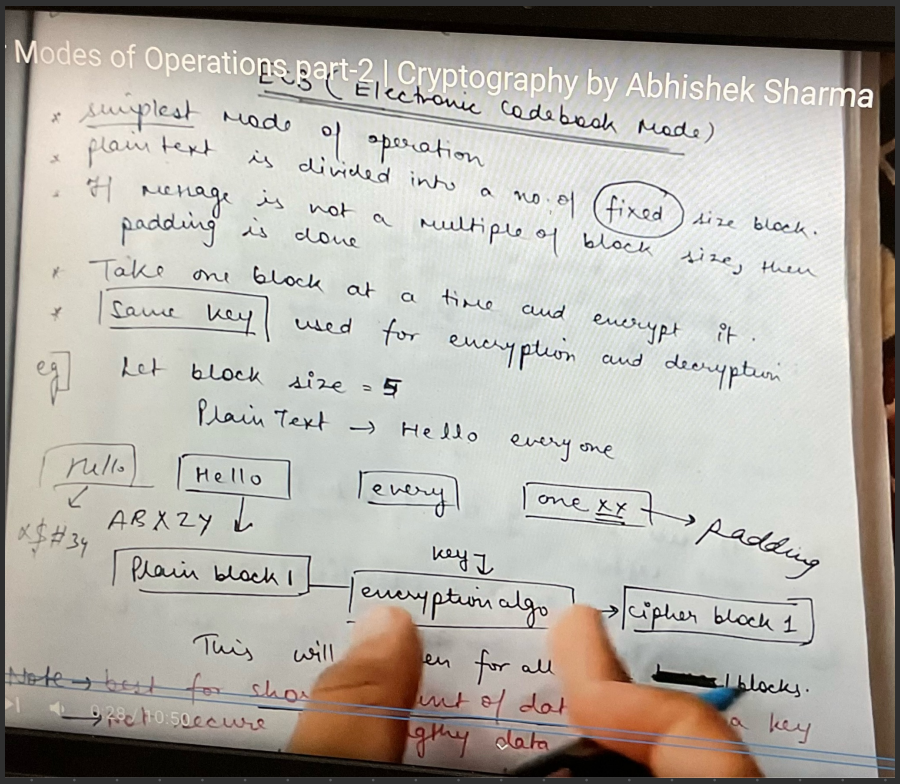
R[0] = 01

R[j] = 02 · R[j-1] ; j=1 ... 9

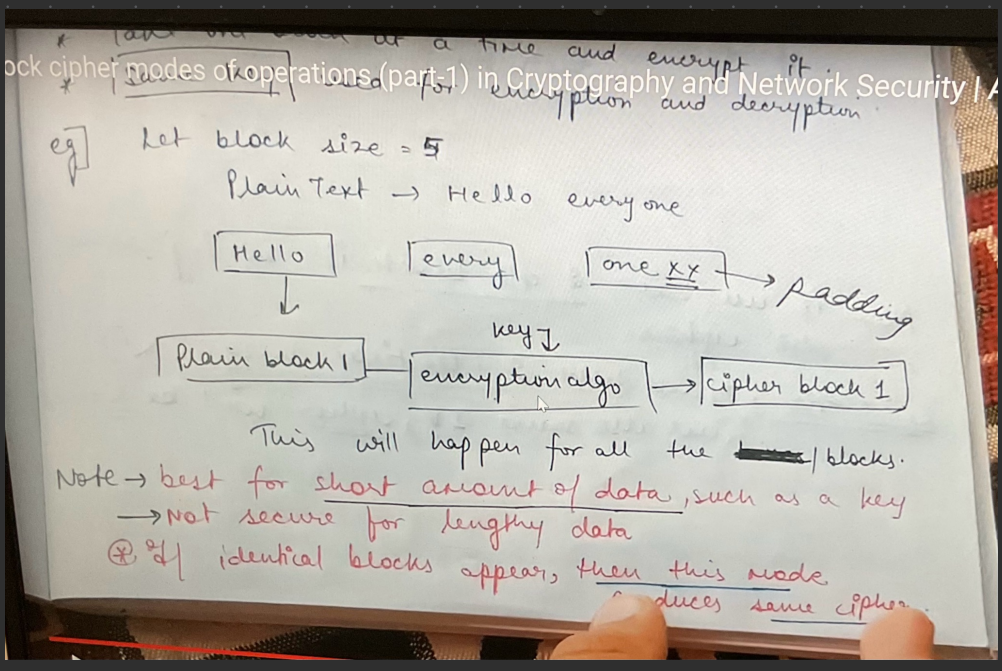
R = [01, 02, 04, 08, 10, 20, 40, 80, 1B, 36]

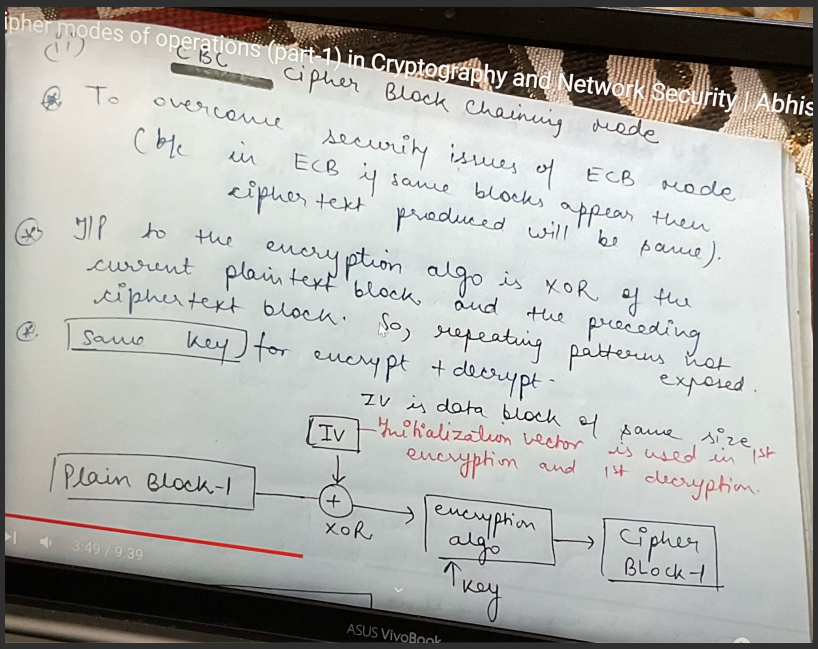
Modes of Operation :



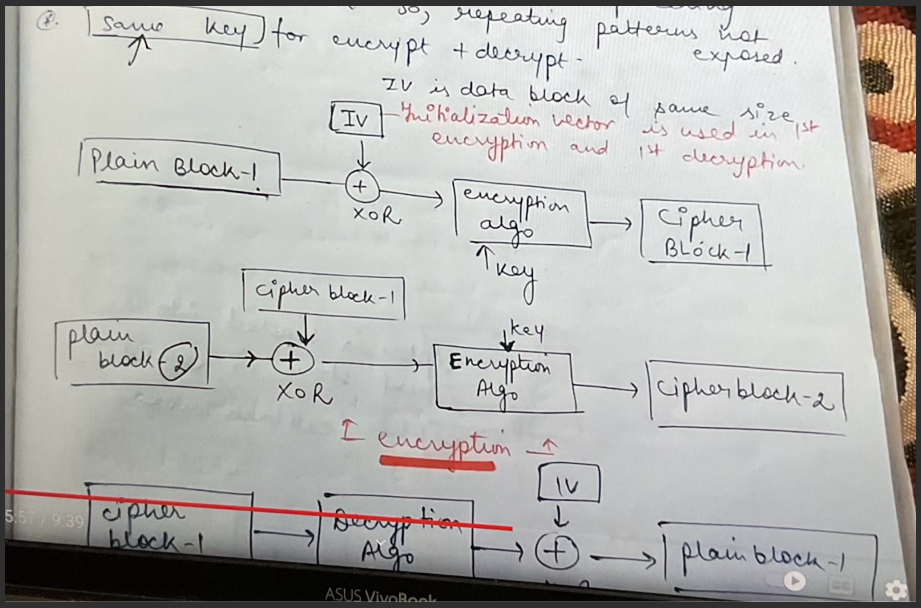


ECB (Electronic Codeback Mode)

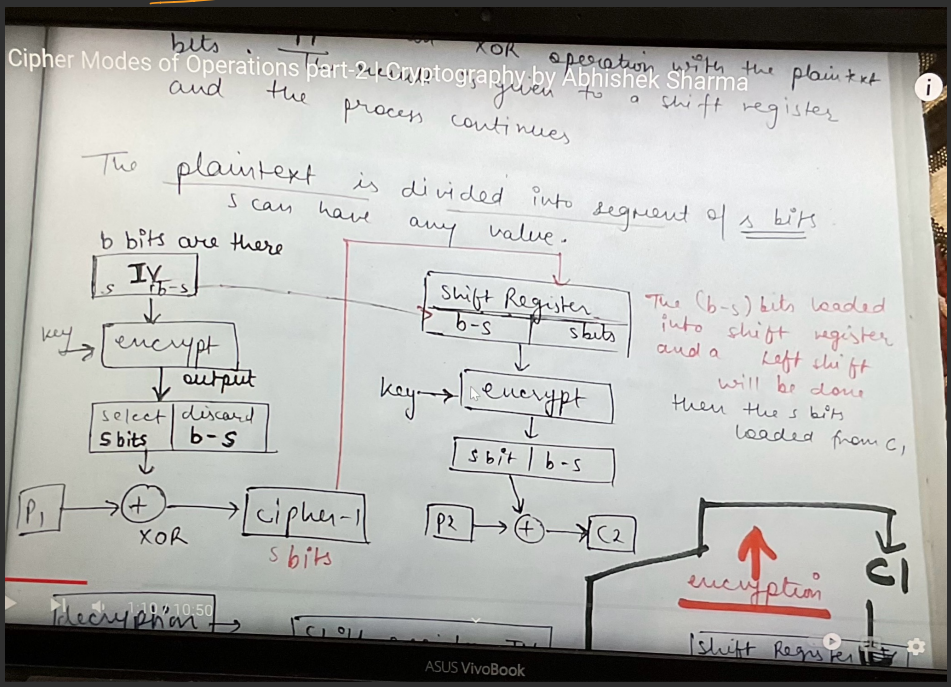




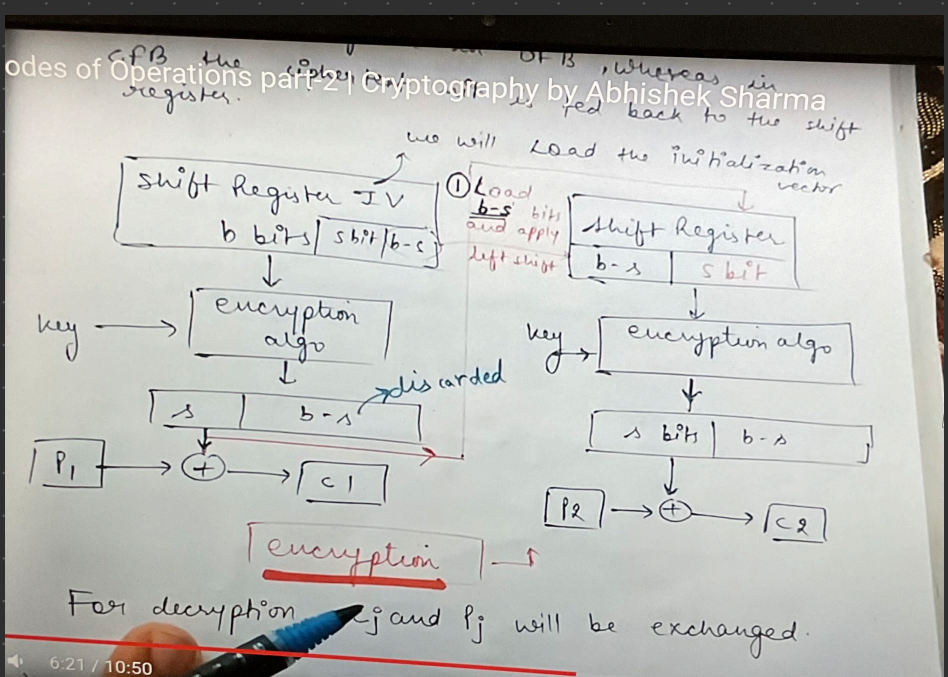
Cipher Block Chaining (CBC)



(Cipher Feedback Mode) CFB –



Output Feedback Mode (OFB) –



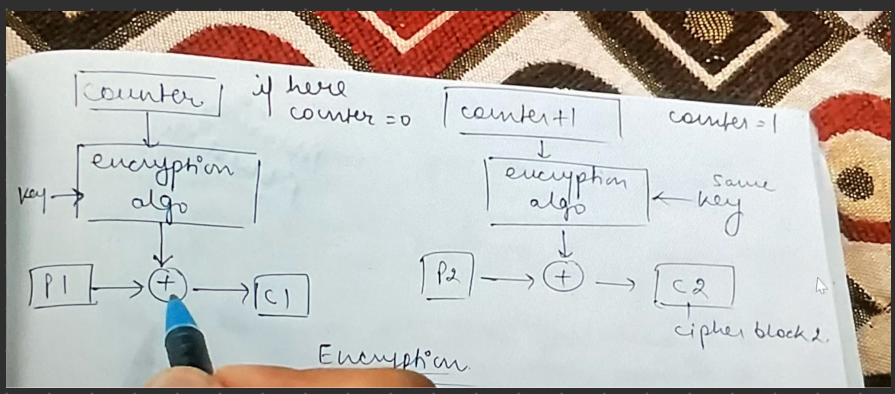
Counter Mode (CTR)

5) COUNTER mode CTR

→ Simple and fast

→ a counter, equal to the plaintext block size is used.

* counter is initialised to some value and then incremented by 1 for each subsequent block.



Mid sem

Diffenential

Public key Cypto

↳ Syntax, Definition

RSA

Elgamal

Discrete log

} William Stalin

Diffie-Hellman

Digital signature

↳ RSA

↳ Elgamal

} Stinson

Linear Cypto.

Hash functions

↳ MACs

mod of -ve integer

$= p - (a \bmod p)$

2 7

Signing Algorithm and Verification

$K \times K^{-1} = 1$

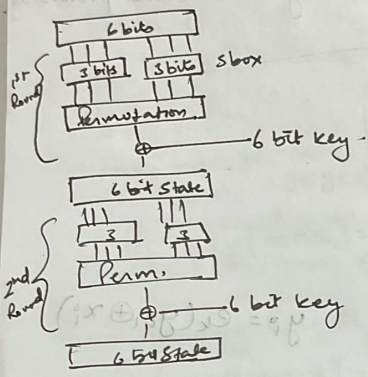
$p - 1$

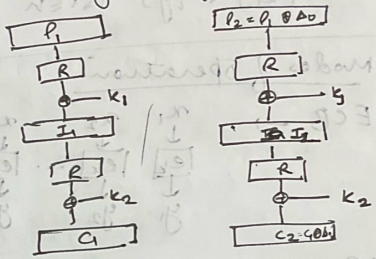
6

14/10/2024

AC 14/10/24

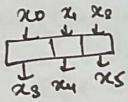
Differential Cryptanalysis





$C_1 = f(x) = P_2(P_1(x))$ (round 2, round 1)

$C_1 = f(x+\Delta) = P_2(P_1(x+\Delta))$



$x_3 = x_0 \oplus x_1 x_2$

$x_4 = x_0 \oplus x_1 \oplus x_0 x_2$

$x_5 = x_0 \oplus x_1 \oplus x_2 \oplus x_0 x_1$

$I_1 = K_1 \oplus R(P_1)$

$I_2 = K_1 \oplus R(P_1 \oplus \Delta_0)$

$I_1 \oplus I_2 = R(P_1) \oplus R(P_1 \oplus \Delta_0)$

→ Input Difference

→ Output Difference.

→ Difference Distribution Table

x_0	x_1	x_2	x_3	x_4	x_5	
0	0	0	0	0	0	→ 0
0	0	1	0	0	1	→ 1
0	1	0	0	1	1	→ 3
0	1	1	1	1	0	→ 6
1	0	0	1	1	1	→ 7
1	0	1	1	0	0	→ 4
1	1	0	1	0	1	→ 5
1	1	1	1	0	1	→ 2

DDT of Sbox

$x \quad x+\Delta_0 \quad y \quad y' \quad y \oplus y'$

17/10/2024

AC 18/10/24 Stinson Sec 4.4

Assume a function $f(x) = f_3(f_2(f_1(x)))$

Assume you find:

- a pair Δ_0, Δ_1 such that:
  $f_1(a + \Delta_0) = f_1(a) \oplus \Delta_1$ with probability 2^{-P_1}
- a Δ_2 such that
  $f_2(b + \Delta_1) = f_2(b) \oplus \Delta_2$ with probability 2^{-P_2}
- a Δ_3 such that
  $f_3(c + \Delta_2) = f_3(c) \oplus \Delta_3$ with probability 2^{-P_3}.

~~And Δ_1~~

And if $f(x) = y$, then $f(x + \Delta_0) = y + \Delta_3$ with probability $2^{-P_1 P_2 P_3}$

Here,

(i) $\Delta_0 \xrightarrow{f} \Delta_3$ is a differential for f.

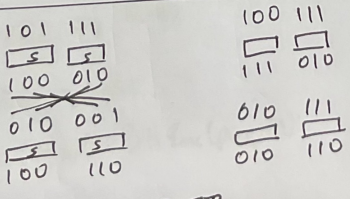
(ii) $\Delta_0 \xrightarrow{f_1} \Delta_1 \xrightarrow{f_2} \Delta_2 \xrightarrow{f_3} \Delta_3$ is a differential trail.

$f(x) = f_n \cdots f_3 f_2 f_1 (x)$

$f(x) = y$

$f(x + \Delta_0) = y + \Delta_0$ with probability $2^{-P_1 P_2 \cdots P_n}$

$\Delta = 001\ 000$

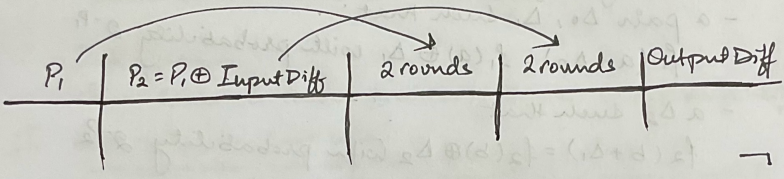


17/10/2024 : cont...

- Let's consider an example with input diff 111

001 000 → ~~000~~ 001 000 ($1/16$)

001 000 → 100 000 ($1/16$)



Differential Distinguisher

Y_T → Maria Eichlseder

H.M. Hayes tutorial sheet

21/10/2024

Public key cryptography (Assymetric) -

- A public key encryption scheme is defined as -

$$\Sigma = (\text{key Gen}, \text{Enc}, \text{Dec})$$

where

① key Gen(1^n) : On input 1^n, generate $sk \leftarrow SK$, $pk \leftarrow PK$ and outputs (sk, pk) as a secret key and Public key pair resp.

② Enc(pk, M) : On input public key pk and message M, outputs $c = Enc_{pk}(M)$ as the ciphertext

③ Dec(sk, c) : on input secret key sk and cipher text c, outputs $M = Dec_{sk}(c)$ as plain text.

End Sem

End Sem Syllabus:

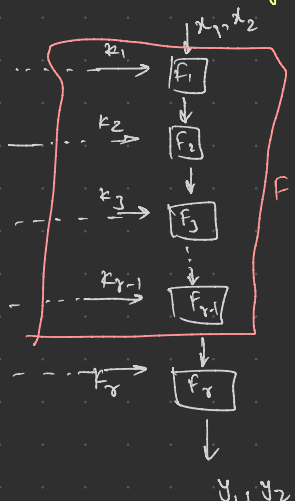
- Differential Cryptanalysis - (Stinsons book 4.4)
-
- Linear Cryptanalysis - Pilling up lemma, linear approximation of s-boxes (Stinsons book 4.3)
- Public Key Cryptography (definition and syntax) - William Stallings book
- RSA - William Stallings book
- Elgamal - William Stallings book
- Number Theory basics - Extended Euclidean Algorithm (Stinson Book - Section 6.2)
- Digital signatures - RSA and Elgamal -Wiliam Stallings book
- Diffie Hellman - William Stalling Book
- Hash functions - Stinsons book 4.1

Differential Cryptanalysis -

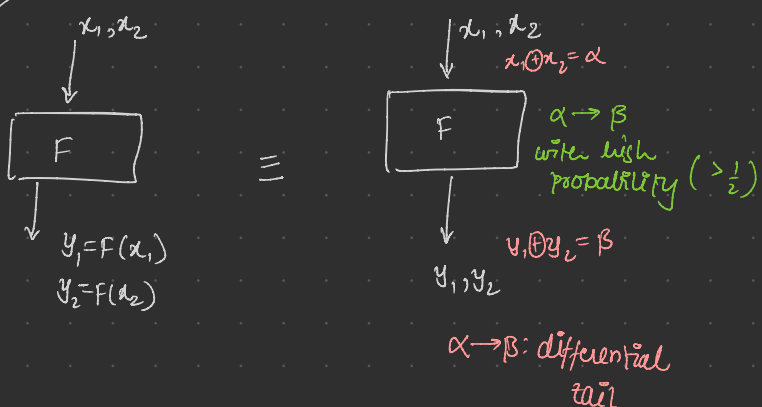
- It is one of the most powerful and widely known cryptanalytic technique used to attack symmetric-key block ciphers.
- It focuses on analyzing how differences in plaintext pairs affect differences in their corresponding ciphertexts.

Definition -

- Differential cryptanalysis operates by taking many pairs of plaintext with fixed XOR differences, and looking at the differences in the resulting ciphertext pairs.
- Based on these difference, probabilities are assigned to possible keys.

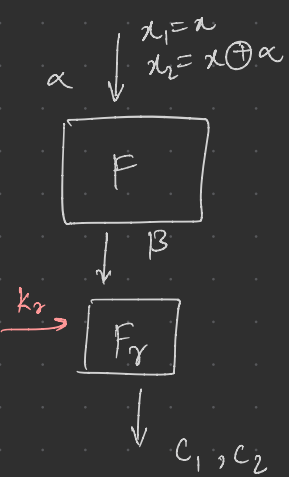


We



- Suppose such $\alpha \to \beta$ exist. Then only we can mount the differential attack.
- DES, FEAL : vulnerable to Diff. attack
- AES : secure

Differential Attack:



$C_1 = Encry(C_1)$

$C_2 = Encry(C_2)$

So, what we have to do is find k_r.

$k_r \rightarrow$	k_r^1	k_r^2	k_r^3	. .	. . .	k_r^i	_ _	_ _ _	_	. .	k_r^N
Freq.	0	0	0	0	0	ϕ_1	0	0	0	0	0

freq initialised with 0.

Now, we do:

$$F_r^{-1}(k_r^i, C_1) = y_1 \quad (\text{any } y_1)$$

$$F_r^{-1}(k_r^i, C_2) = y_2 \quad (\text{any } y_2)$$

and if this $y_1 \oplus y_2 = \beta$

then we increase the freq. of k_r^i.

$\Rightarrow$ We do this process to other pairs of the plain text and their corresponding cipher text.

At the end, the k_r^i whose freq. is most will be the most probably our k_r.

Linear cryptanalysis :

- It is a form of known-plaintext attack where the attacker exploits linear approximations of the cryptographic transformation (such as S-box in a block cipher).

- It relies on finding linear relationships b/w plaintexts, their corresponding ciphertexts and key bits that hold with some probability.

- The idea is to use a set of known plaintexts and their corresponding ciphertexts to guess key bits by maximizing the correlation b/w certain linear expressions.

Definition :

- For a random variable having (hexadecimal) input sum a and output sum b (where $a = (a_1, a_2, a_3, a_4)$ and $b = (b_1, b_2, b_3, b_4)$, in binary)

  let $N_L(a,b)$ denote the number of binary eight-tuples $(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4)$

  such that

$$y_1, y_2, y_3, y_4 = \pi_s(x_1, x_2, x_3, x_4)$$

  and

$$\left(\bigoplus_{i=1}^{4} a_i x_i\right) \oplus \left(\bigoplus_{i=1}^{4} b_i y_i\right) = 0$$

Pilling-up lemma :

- Pilling-up lemma is a key technique in linear cryptanalysis that allows attackers to combine multiple linear approximations in order to increase the probability of successfully guessing key bits.

- The lemma states that if you have multiple linear approximations with known probabilities, the overall correlation can be approximated by multiplying these individual probabilities.

Let $X_1, X_2, X_3, \ldots, X_i$ are independent random variable-

$$P(X_i = 0) = p_i$$

and $$P(X_i = 1) = 1 - p_i$$

bias of X_i : $\epsilon_i = p_i - \frac{1}{2}$

$$\therefore P(X_i = 0) = \epsilon_i + \frac{1}{2}$$

$$P(X_i = 1) = \frac{1}{2} - \epsilon_i$$

We want to know the bias of this-

$$\left(\bigoplus_{i=1}^{m} a_i x_i\right) \oplus \left(\bigoplus_{i=1}^{n} b_i y_i\right)$$

and if the bias is $\neq 0$ then we can say there is linear relationship.

For (X_i, X_j) ;

$$P(X_i = 0, X_j = 0) = p_i p_j \quad -(1)$$

$$P(X_i = 0, X_j = 1) = p_i (1 - p_j) \quad -(2)$$

$$P(X_i = 1, X_j = 0) = (1 - p_i) p_j \quad -(3)$$

$$P(X_i = 1, X_j = 1) = (1 - p_i)(1 - p_j) \quad -(4)$$

$\Rightarrow X_i \oplus X_j = (X_i + X_j) \bmod 2$

$P(X_i \oplus X_j = 0) = P_i P_j + (1-P_i)(1-P_j)$ using ① & ④

and $P(X_i \oplus X_j = 1) = P_i(1-P_j) + (1-P_i)P_j$

bias : $\boxed{\epsilon_{i,j} = 2\,\epsilon_i\,\epsilon_j}$

The general version of this is piling-up lemma -

$$\epsilon_{i_1, i_2, i_3, \ldots, i_k} = 2^{k-1} \cdot \prod_{j=1}^{k} \epsilon_{i_j}$$

Note → if $\epsilon_{i_j} = 0 \quad \forall j$ then $\epsilon_{i_1, i_2, i_3, \ldots i_k} = 0$

and also for $X_{i_1} \oplus X_{i_2} \oplus X_{i_3} \oplus \cdots \oplus X_{i_j}$, bias $\epsilon_{i_1, i_2, \ldots i_k} = 0$

Linear Approximation of S-box :

- An s-box is a key component in many block ciphers and linear cryptanalysis works by finding linear approximation of the s-boxes.

- The goal is to express the s-box as a linear function, or find a linear relation between the input and output bits of the s-box.

- A linear approx^n is a bitwise linear equ^n of the form:

$$P = \bigoplus_{i=1}^{m} x_i \bigoplus_{j=1}^{n} y_j \oplus k$$

where P = output (ciphertext), x_i are input bits
y_i are output bits
and k is the key.

Now, if the bias of the above equation is not zero, then there would be a linear relationship exists.

Public key Cryptography -

Public key cryptography (Assymetric) -

- A public key encryption scheme is defined as -

$$\Sigma = (\text{key Gen}, \text{Enc}, \text{Dec})$$

where

① key Gen(1^n) : On input 1^n, generate $sk \leftarrow SK$, $pk \leftarrow PK$ and outputs (sk, pk) as a secret key and Public key pair resp.

② Enc(pk, M) : On input public key pk and message M, outputs $c = Enc_{pk}(M)$ as the ciphertext

③ Dec(sk, c) : on input secret key sk and cipher text c, outputs $M = Dec_{sk}(c)$ as plain text.

RSA - 1977 (Ron Rivest, Adi Shamir, Len Adleman)

$1 \bmod n = nk + 1$

Enc → $c = M^e \bmod(n)$

Dec → $c^d \bmod(n)$

$M^{de} \bmod(n) = M \bmod(n)$

$a^{p-1} = 1 \bmod n$

$a^p = a \bmod n$

here, $M < n$

↳ d and e are multiplicative inverses of modulo $\phi(n)$.

i.e., $de = 1 \bmod \phi(n)$

$d = e^{-1} \bmod \phi(n)$

Key generation
→ Choose n
→ choose $e < n$
→ compute $\phi(n)$, d

Public key → (e, n)

secret key → (d, n)

⇒ Properties in Modular Arithmetic :

① if $ab = ac \bmod n$

$\Rightarrow b = c \bmod n$, if a is relatively prime to n.

② a number 'P' is called a prime iff they are divisible by ± 1 and $\pm P$.

③ Fermat's Theorem -

if P is prime and a is a positive integer not divisible by P

then, $a^{P-1} = 1 \bmod P$

Or $a^P = a \bmod P$

④ Euler's Totent / Phi function :

This function $\phi(n)$. is the number of positive integers less than n and relative prime to n.

Note: $\phi(1) = 1$ and $\phi(P) = P-1$, if P is prime

Example: $\phi(7) = 6$

$\phi(6) = 2$ (i.e., only 1, 5 are co-prime to 6)

⑤ Euler's theorem :

for every a and n

$$a^{\phi(n)} = 1 \bmod n$$

or $a^{\phi(n)+1} = a \bmod n$

⑥ Important Property :

if p and q are prime

$$\boxed{\phi(p \cdot q) = (p-1)(q-1)}$$

so, if $n = pq$, then $\phi(n) = (p-1)(q-1)$

Enc : $C \longrightarrow M^e \bmod (n)$

Dec : $M \longrightarrow C^d \bmod (n)$

Public : (e, n)

Secret : (d, n)

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

$$M^{ed} = M \bmod n \Longrightarrow \text{By Euler's theorem}$$

$$a^{k\phi(n)+1} = a \bmod n$$

$ed = k\phi(n) + 1$

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

$x = kn + 1$

$x = 1 \bmod (n)$

$e = d^{-1} \bmod \phi(n)$

$d = e^{-1} \bmod \phi(n)$

now,

Choose any $n = 11$

Choose $e = 9$

take $M = 8$

find $C = ?$

RSA

① Key Generation

(i) Choose p and q where p & q are prime

(ii) compute n & $\phi(n)$

(iii) choose e

(iv) compute $d = e^{-1} \bmod \phi(n)$

public key : (n, e)

secret key : (d, p, q)

② Encryption function

$Enc(e, n, M) = C = M^e \bmod n$

③ Decryption function

$Dec(d, c, n) = M = C^d \bmod n$

Example (2.2 page 49)

① key Generation:

(i) choose prime, p & q = 17 & 11

(ii) compute $n = 17 * 11 = 187$

$\phi(n) = 16 \times 10 = 160$

(iii) choose e relatively prime to $\phi(n)$, i.e, $gcd(e, \phi(n)) = 1$

(iv) compute d such that $de = 1 \bmod \phi(n)$

$d * 7 = 160 + 1 \Rightarrow \boxed{d = 23}$

Public key: $(e, n) = (7, 187)$

secret key : $(d, p, q) = (23, 17, 11)$

② Encryption:

let $M = 88$

$C = M^e \bmod n$

$= 88^7 \bmod 187$

$= 88 * 77^2$

$= 88 * 77 * 132$

$\Rightarrow \boxed{C = 11}$

$a^{x+y+z} \bmod n$

$= (a^x \bmod n * a^y \bmod n * a^z \bmod n)$

③ Decryption:

$C = 11$

$M = 11^{23} \mod 187$

$= 11^2 * 11^4 * 11^8 * 11^8 * 11 \mod 187$

$= 88$

Extended Euclidean:

a, b such that $\gcd(a, b) = d$

then we have x, y such that $ax + by = d = \gcd(a, b)$

$ed, \phi(n)$; $\gcd(ed, \phi(n)) = 1$

x, y such that $ed \cdot x + \phi(n) \cdot y = 1$

Digital signature (ElGamal)

Let P be a prime number and α be its primitive root.

Let $K = \{(P, \alpha, a, \beta) : \beta = \alpha^a \bmod P\}$, K can't be 0.
(a: secret key)

Public key : P, α, β

Private key : a

Signing Algorithm :

Choose a random number $k \in Z^*_{P-1}$ → one element less i.e., 0

define :

$$Sig_k(x, k) = (r, \delta)$$; x is the plain text that you want to sign

where :

$$r = \alpha^k \bmod P$$

and $$\delta = (x - ar)\, k^{-1} \bmod (P-1)$$

Verification Algorithm :

$Ver_k(x, (r, \delta))$ = true iff $\beta^r r^\delta = \alpha^x \bmod p$

⇒ Algamal uses Digital signature

Elgamal

P, α, a

P, α : Public

a : Secret random number

$\alpha^a \bmod P$ → Public

Public key → P, α and $\alpha^a = \beta$

where private key → a

Choose a random number k

→ encryption has two steps–

$$y_1 = \alpha^k \bmod P$$

$$y_2 = M \beta^k \text{ and } P \text{ ; for msg } M$$

→ Decryption :

$$y_2 (y_1^a)^{-1} \bmod P.$$

Exercise :

Let $P = 467$, $\alpha = 2$, $a = 127$

Let $x = 100$, and assume $K = 213$

Compute the signature using Elgamal s.s and verify its connection.

K^{-1} = inverse of K modulo $(P-1)$

$K^{-1} = 431$

⇒ Signature Computation

$Sig(x, K) = Sig(100, 213)$

$r = 2^{213} \bmod 467$ ($\alpha^K \bmod p$)

$= 29$

$\delta = (x - ar)\, K^{-1} \bmod (P-1)$

$= (100 - 127 * 29)\, 431 \bmod 466$

$= -3583 * 431 \bmod 466$

$= -1544273 \bmod 466$

$= 51$

⇒ Verification

$$\beta^r r^\alpha = \alpha^x$$

Elgamal :

- Elgamal is a public-key cryptosystem that uses modular arithmetic and is based on the Discrete Logarithm Problem.
- It allows for secure encryption and decryption using a private key for decryption and a public key for encryption.

→ Elgamal Encryption Process :-

here α, p are global elements

① Key Generation (by the receiver)

(i) Select a large prime number (p)

(ii) Select α ; $\alpha < q$ and α is a primitive root of p.

(iii) Select X_A : private key where $X_A < p-1$

(iv) Calculate Y_A (or say β) : $\beta = \alpha^{X_A} \bmod p$

Now, Public key : $\{p, \alpha, \beta\}$

Private key : X_A

② Message Encryption (by sender) :

Plain text : $M < p$

(i) Choose a random integer R.

(ii) Calculate one time key $K = \beta^R \bmod p$

(iii) CipherText : (C_1, C_2)

where $C_1 = \alpha^R \bmod p$

$C_2 = K \cdot M \bmod p$

③ Message Decryption (by the Receiver) - Ciphertext : (C_1, C_2)

(i) Calculate $K = (C_1)^{X_A} \mod P$

(ii) Plaintext $M = (C_2 K^{-1}) \mod P$

Why is Elgamal Secure?

The security of Elgamal is based on the difficulty of solving the **Discrete Logarithm Problem** (DLP), which is the task of finding the exponent X_A given $a^{X_A} \mod q$. As long as the numbers used (like q) are large enough, it is computationally infeasible for an attacker to determine the private key X_A or recover the plaintext M from the ciphertext without knowing the private key.

→ Key Advantage:

(i) Probabilistic encryption : involves randomness

(ii) ↑ Security

→ Key Disadvantage:

(i) Larger Cipher Text size

(ii) ↓ Performance due to comput^{n} of modular exponentiations

Diffie-Hellman key exchange :

- not an encryption algorithm, but a key exchanging algorithm.

- The purpose of the algorithm is to enable two users to securely exchange a key that can be then used for subsequent symmetric encryption of messages.

Algorithm

(i) Consider a prime no. q

(ii) Select α, $\alpha < q$ where α is a primitive root of q.

α, q : Gobal components

(iii) Assume X_A : Private key of person A known to only A $(X_A < q)$

→ Calculate $Y_A = \alpha^{X_A} \bmod q$

↳ Public key of A

(iv) Assume X_B : private key of person B known to only B $(X_B < q)$

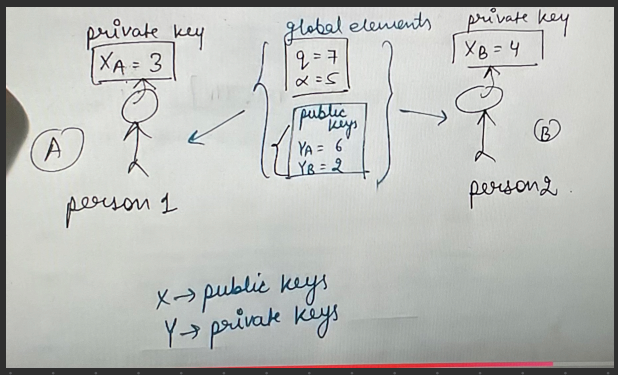
→ Calculate $Y_B = \alpha^{X_B} \bmod q$

↳ Public key of B

Note

Both person A and B doesn't need any information from anyone. They just use α and q which is known to everyone.

example:



(v) Now, we will calculate the secret key

- Both the sender & Receiver will use public key of the other one.

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

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

Public key known to all

if $K_A = K_B$ then we can say exchange is successful.

Hash Function :

- A hash function is a mathematical algorithm that transforms an input (or message) of arbitrary length into a fixed-length output, typically represented as a string of characters or a number, often called the hash value or digest.

Definition -

- can be defined using 4 tuples -

(X, Y, K, H)

X → input, Y → output, K → keyset, H → Hash function

- A hash family is a four tuple, where -

(i) X is a set of possible messages ($x \in X$ is of unknown length)

(ii) Y is a set of "hash values" / "message digest" ($y \in Y$ is of fixed length)

(iii) K is a set of possible keys.

(iv) for every $k \in K$, there exist a hash function $h_k \in H$

$$h : X \longrightarrow Y$$

- A pair $(x, y) \in (X, Y)$ is said to be a valid pair if

$$h_k(x) = y$$

Hash function Property –

1. There should not exist x, x' such that $x \neq x'$ but $h(x) = h(x')$
2. $h^{-1}(y) = x$ | pre-image
3. No collision before exhaustion of x

Security of hash function –

- Let $h : X \longrightarrow Y$ be an unkeyed hash function, h is said to be secure if the following 3 problems are difficult to solve for h:

① Pre-image Problem:

- instance : A hash $h : X \longrightarrow Y$ and an element $y \in Y$
- find $x \in X$ such that $h(x) = y$

→ A hash function for which finding x (i.e., pre-image) is not efficient is called pre-image resistant.

Pseudo code

$|Y| = B$

choose any $X_0 \subseteq X$, $|X_0| = A$

for each $x \in X_0$

do

{ if $h(x) = y$ then return x

return (fail)

if $A < B$: h is insecure

if $B = A$: h is secure

② Second Preimage Problem –

- instance : A hash $h : X \longrightarrow Y$ & an element $x \in X$
- find : $x' \in X$ such that $h(x) = h(x')$

A hash function for which finding x' (i.e., second pre-image) is not efficient is called second pre-image resistant.

Pseudo code

$y \leftarrow h(x)$, $|Y| = B$

choose $X_0 \subseteq Y \setminus \{x\}$, $|X_0| = A$

for each $x' \in X_0$

do { if $h(x') = y$

return x'

return (fail)

if $A < B$: h is insecure

if $A = B$: h is secure

③ Collision :

- instance : A hash function $h : X \longrightarrow Y$
- find : $x, x' \in X$ such that $h(x) = h(x')$ where $x \neq x'$

A hash function for which collision can't be found efficiently is said to be collision resistant.

$|Y| = B$

Pseudo code

choose $X_0 \subseteq X$, $|X_0| = A$

for each $x \in X_0$

do $y_x \leftarrow h(x)$

if $y_x = y_{x'}$ for some $x' \neq x$

then return (x, x')

else return (failure)

if $A < B$: h is insecure

if $A = B$: h is secure