D : P + C

C : K + C

Date

: Key Schedule Algo

① one sub key for each round

② select subkeys to maximize difficulty of deducing individual subkeys & difficulty of working back to main key.

③ key schedule algo should guarantee Strict Avalanche & Bit Independence criterion.

Sessional - II

Chapter 6 Advanced Encryption Standard.

Finite Field Arithmetic

- 128, 192, 256 bit block size ← variants
- + key size
- ops. performed on 8-bit bytes
  break block into 8-bits
- arithmetic ops. performed over finite field $GF(2^8)$
- field is a set in which we can do +, ÷, ×, − without leaving set

  2^8 set $\{0, 1, 2 \ldots 255\}$ → result should be within this range

- Division: $a/b = a(b^{-1})$
- to keep values/result within predefined range, take mod with 26 (like Caesar cipher)

Example:

$a = x^2 + x + 1$ (7, 0111b), $b = x + 1$ (3, 0011b) with

0	1	1	1

$0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$

$0 + x^2 + x^1 + x^0$

$x^2 + x + 1$

0	0	1	1

$x^1 \; x^0$

$1 \cdot x^1 + 1 \cdot x^0$

$x + 1$

primitive of $x^4 + x + 1$ ($GF(2^4)$).

result has → x^3 i.e. → 4 bits then divide result by primitive to normalize it.

• primitives are non-reducable polynomials.
so we use them to do normalization

Date

Addition $(x^2 + x + 1) + (x+1)$

$= x^2 + 2x + 2$

2 mod 2 = 0

not possible cuz binary
so take mod with 2 to remove it

$= x^2 + 0.x + 0$

$= x^2$ ⟶ $x^3\ x^2\ x^1\ x^0$: 0 1 0 0 = 4

Multiplication $(x^2 + x + 1) \times (x+1)$

$= x^3 + x^2 + x + x^2 + x + 1$

$= x^3 + 2x^2 + 2x + 1$

$= x^3 + 0.x^2 + 0.x + 1$

$= x^3 + 1$ ⟶ $x^3\ x^2\ x^1\ x^0$: 1 0 0 1 ⟶ 9

Example:

primitive = $x^4 + x + 1$

$a = x^3$ $\quad b = x^2 + 1$

$x^3 + x^2 + 1$ ⟶ $x^5\ x^4\ x^3\ x^2\ x^1\ x^0$: 1 0 1 0 0 0 ⟶ 6-bits

$= x^3 \times (x^2 + 1)$

$= x^5 + x^3$ (overflow)

divide by primitive

$$x^4 + x + 1 \;\overline{)\; x^5 + x^3 \;(\; x}$$

$x^5 + x^2 + x$

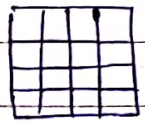
$2x^5 + x^3 + x^2 + x$

$0.x^5$

$x^3 + x^2 + x$ ⟵ 1110 (normalized)

Date

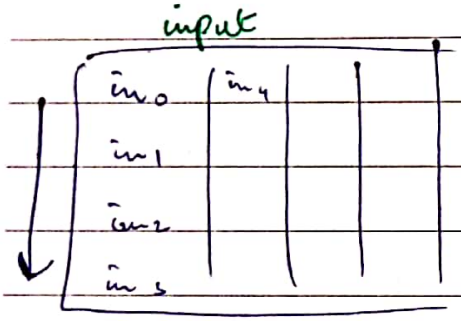
AES Encryption Process



16B, 128b

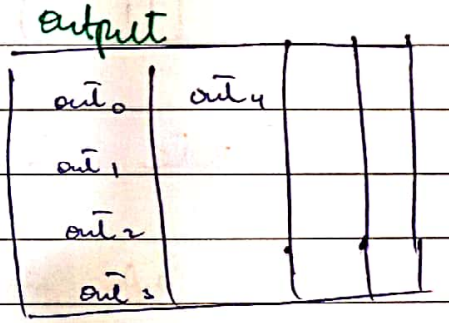
- Input state : every box will contain 1 byte
- Initial transformation (Round 0) : involves key
- we get state matrix after every round
- Round 1 to N-1 : 4 transformations
  Round N : 3 transformations
- 16B key size for all rounds
- Ciphertext : 16B, 128b
- even when key size increases from 128b, the size of PT, CT stays same.

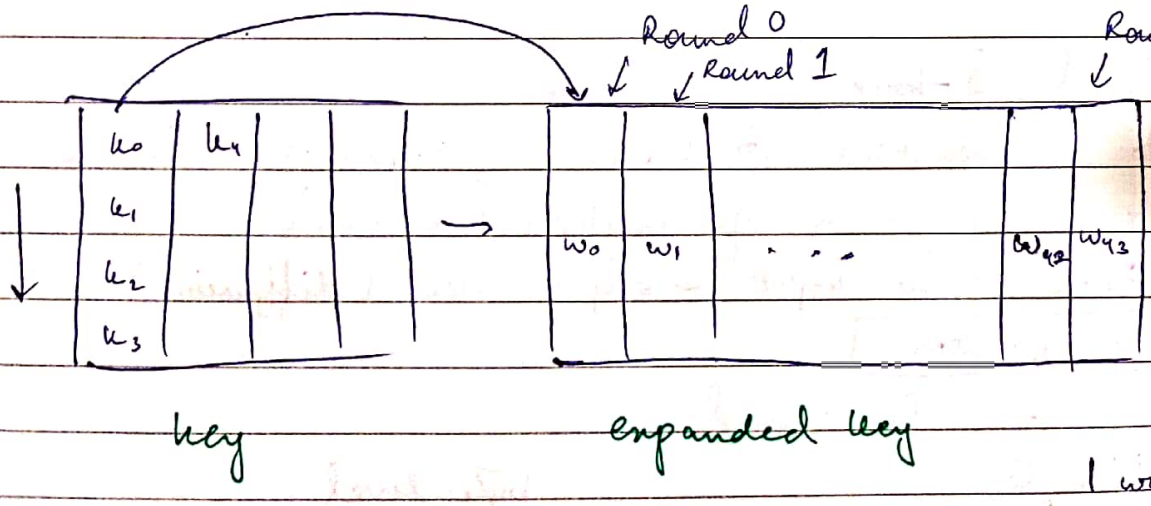
no. of rounds	key size
10	16
12	24
14	32



state array

in slide





- 44 words if 128b key size
- fixed
- 4B in 1 word
  colum

1 word – 4B

44 words – 11 rounds

4 B

- key size in a round = 4 words

Date

- why use finite fields?
- diffusion does low correlation b/w P + C.

AES Parameters
(in slides)

November 1st, 2021.

AES Byte Level Operations.

contributing to Diffusion

Substitution:

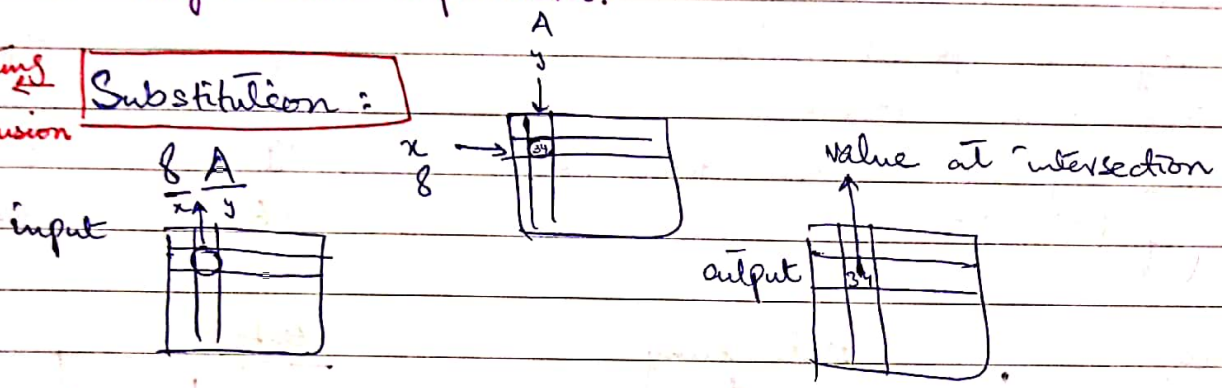


Table 6.2 S-box → fixed matrix

↑

8A → 7E Inverse S-box

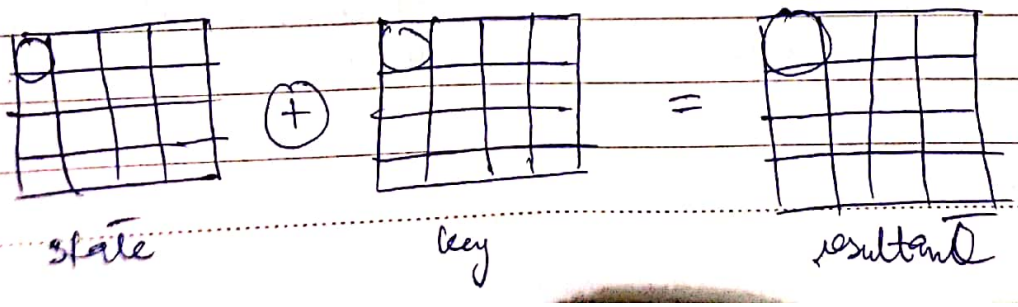
05 → 6B

- Rational behind S-box
  - designed to be resistant to known cryptoanalytic attacks.
  - nonlinearity is due to use of multiplicative inverse
  - low correlation b/w input & output bits (diffusion)

Add Round Key Transformation:

4 words key for each round

1 word = 4 bytes byte level

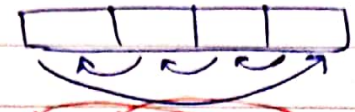


Date circular

Shift row T: left byte level shift

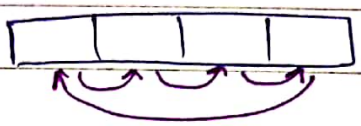
4×4 matrix 1st row : as it is

2nd row : 1 shift



3rd row : 2 shift

4th row : 3 shift



Shift Row Rationale:

- initial values of each column are spread out in whole matrix
- distance of multiple of 4-bytes because col. change.

Inverse Shift row T:

byte level right shift

Mix Column T:

$$\begin{bmatrix} 4\times4 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ 4\times4 \end{bmatrix} = \begin{bmatrix} 4\times4 \end{bmatrix}$$

fixed — State matrix

in fixed matrix, here's only 1, 2, 3 → conditional bitwise XOR

addition/XOR

$$S'_{0,j} = (2 \times S_{0,j}) \oplus (3 \times S_{1,j}) \oplus (01 \times S_{2,j}) \oplus (1 \times S_{3,j})$$

$S'_{1,j} =$

$S'_{2,j} =$

$S'_{3,j} =$

* conditional bitwise XOR with 0001 1011 if leftmost bit of original value before shift is 1

e.g. 03×6E

1 - shift left

2 - do XOR

Date

Example:

if 03 × 6E

↓

find binary of 6E and do XOR.

Then do shift left

Inverse Mix Column

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ & & & \end{bmatrix} \cdot \begin{bmatrix} 4\times4 \\ \text{State matrix} \end{bmatrix} = \begin{bmatrix} 4\times4 \end{bmatrix}$$

4 x 4 fixed

↓

matrix x fixed matrix on prev. page = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

if identify matrix then

correct inverse of each of

---

2 7

1000 0111

0000 1111

0001 1110

174

1010 1110

0101 1101

6 E

0110 1110

1101 1100

0001 1011

1100 0111

128 64 32 16 8 4 2 1

0 1 0 1 0 1 1 1

1 0 1 0 1 1 1 0

00 00 00 10 02

x

0000 1000 08

x^3 x^2 x^1 x^0

0000 1010

Date November 2nd, 2021

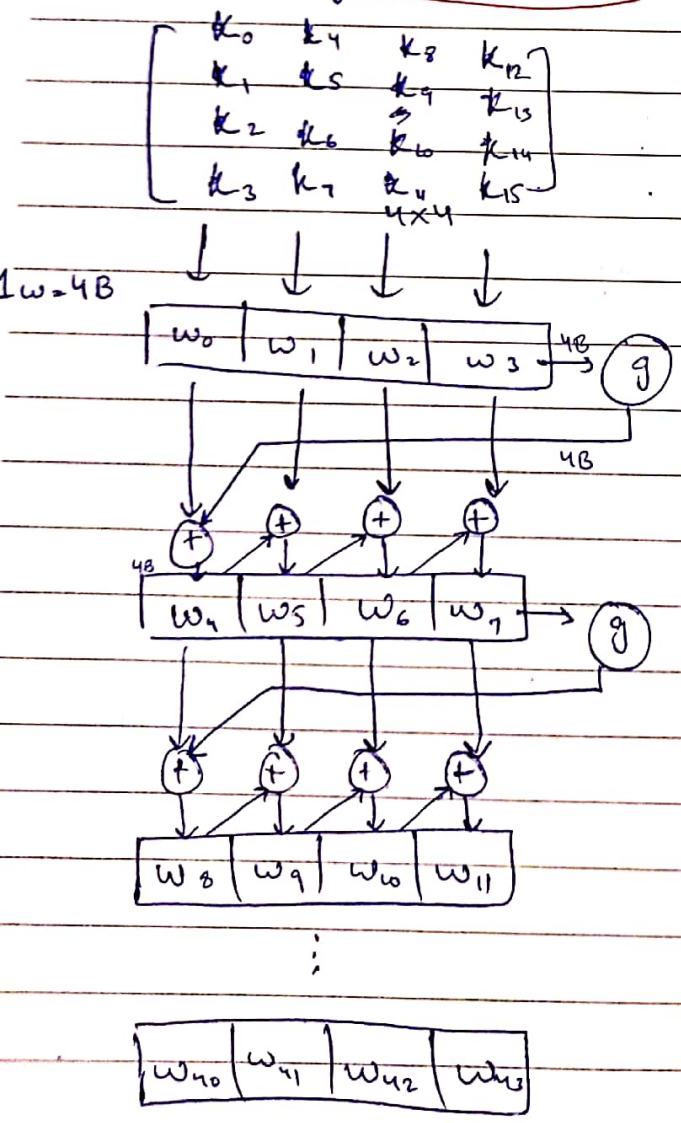
AddRoundKey Transformation:

- 128 bits of State are bitwise XORed with 128 bits of round key
- is as simple as possible + affects every bit of State

:

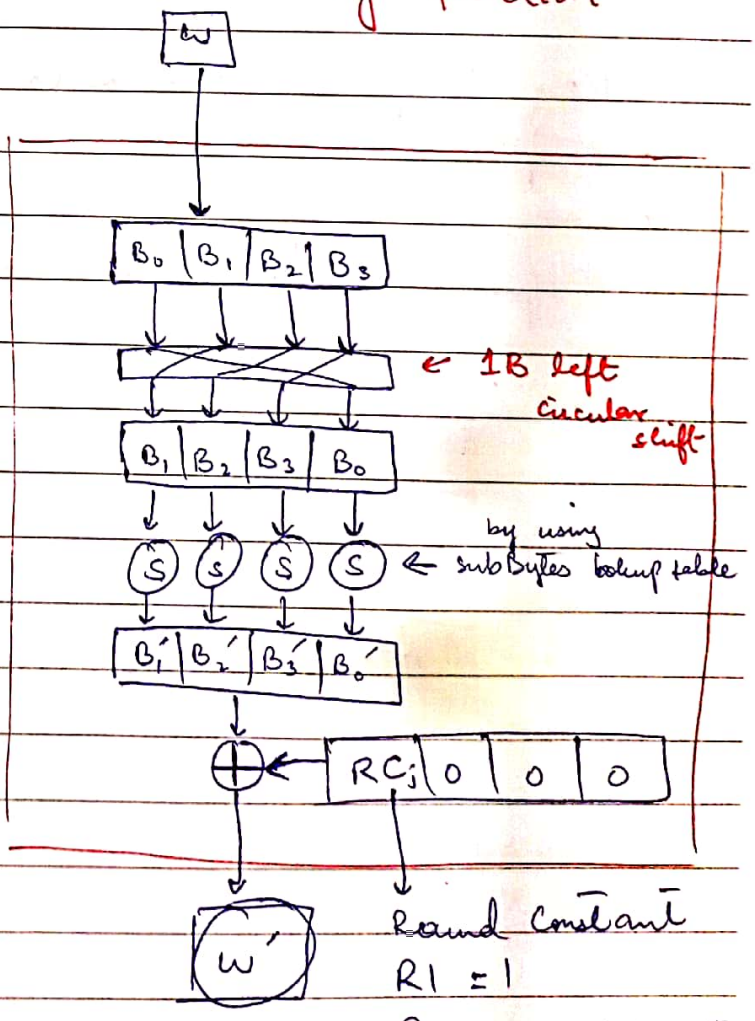
- Constant stuff : ① Substitution box, inverse Sub. box
  ② fixed matrix in MixCols. / inverse mix cols.
- Variable stuff : ① round key for every stage

AES Key Expansion



44 words to complete 10 rounds

g function



Round Constant

R1 = 1

R2 = multiply with hexa 2.

RC[j] = 0x02 × RC[j−1]

Date

10 rounds + additional Round
11 keys required of 16B

j	4	5	→ rest in slides
RC[j]	08	10 (hexa)	

0x00

Rcon[i] = (RC[i], 0, 0, 0)

↓ Round constant RC

0000 0010 02

0000 1000 08

x x^3

$x \cdot x^3 = x^4$

0001 0000 x^4

hexa: 1 0

RC[1] = 0x01

RC[j] = 0x02 × RC[j-1].

Key Expansion Rationale:

- ~~Confusion/diffusion~~
- non-linearity
- diffusion of cipher keys into round keys: wide spread

Table 6.3 example.

Table 6.4

Table 6.5 Avalanch Effect in AES.

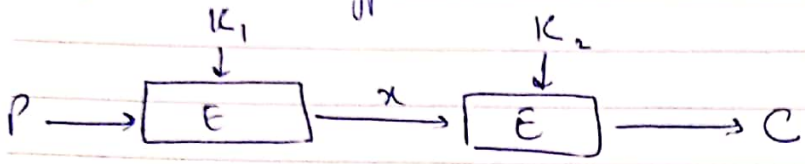
Round Plain Text No. of bits

Lecture #06

Date

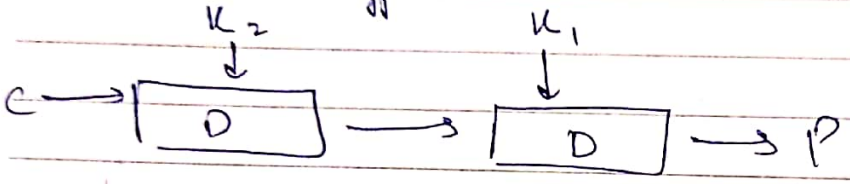
Block Cipher Ops.

Double Encryption



$$C = E(K_2, (E(K_1, P)))$$

Double Decryption



$$P = D(K_1, (D(K_2, C)))$$

DES

Prior key search space: 2^{56}

Now " $: 2^{56} \cdot 2^{56}$

$= 2^{112}$

Meet-in-The-Middle Attack

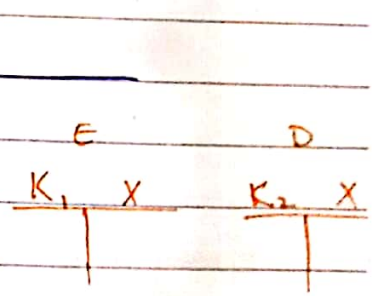
$$X = E(K_1, P) = D(K_2, C)$$

Given a pair of (P, C)

E
1. Encrypt P for all 2^{56} values of K_1
2. Store results & sort them

D
3. Decrypt C using all 2^{56} values of K_2. After each description check vals in table.
4. If match found then Test K_1, K_2 pair on another P-C sample to verify

$\rightarrow (2^{56} X)$



↓

no need as we have X lookup table

Vulnerable Double D/E so use AES not Double D/E DES.

K_1 K_2

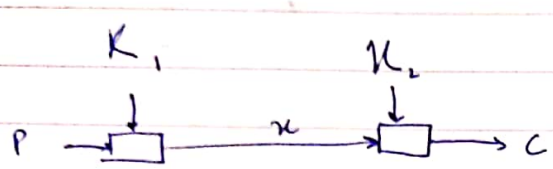
2^{56} 2^{56} $2 \times 2^{56} = 2^{57}$

Search space

Date November 8th, 2021. Morning Arif Dar

Double Encryption DES.

$2^{56} \cdot 2^{56} = 2^{112}$



56 key size of K_1 ... 000 ... 000 (55)

K_2 ... 00 ... 101 (55)

check it with all keys of K_2

* Double E. in DES not much good as search space 2^{57} which is close to 2^{56}.

Example: Man in the middle $= 2^{57}$

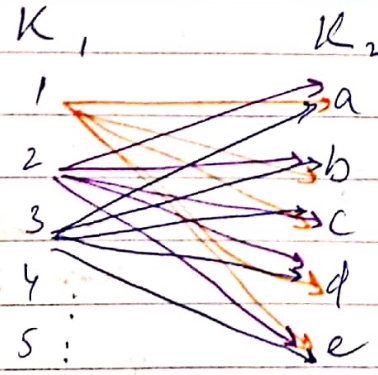
K_1	K_2
1	a
2	b
3	c
4	d
5	e

K_1	X
1	$
2	&
3	*
4	?
5	/

X	K_2
–	a
$	b
	c
	d
	e

→ not found in K_1 X table so we know 'a' is not correct key

Example: Double E/D DES 2^{112} = 10



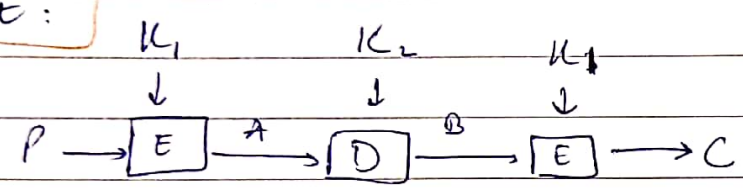
5 + 5 + 5 + 5 + 5 = 25.

Date

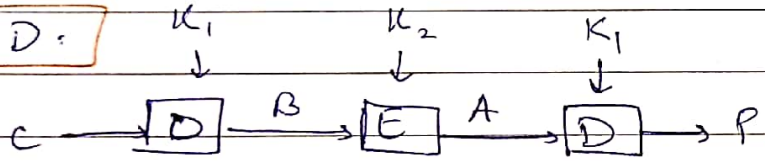
Triple - DES w/ Two Keys

- counter to man middle attack is to use 3 stages w/ 3 diffent keys
  ↓
  raises cost of attack to 2^{112} — impractical
- alternate : Triple DES w/ 2 keys.

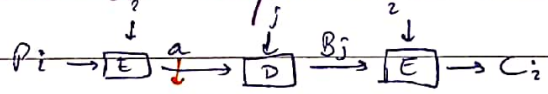
E:



D:



Man in Middle Attack on Triple DES:



Pi Ci

0 . .

1 . .

. . .

. . .

n-1 . .

n entries

Assignment : ① take DES algo in Python Builtin / algo

② apply key 2 times, 2 fn calls.

④ emulate MitMA

③ apply DES 1 time and keep (P, C) pair with you.

⑤ make lookup table after apply 2^{56} keys and store it

⑥ sort

⑦ take 1 K_2 and check in lookup table

⑧ find avg. for 2, 3 pairs

$0 \leq avg \leq 2^{57}$

make word doc if some issue and add screenshots

⑨ ½ time as well

Date November 9th, 2021.

Triple E/D w/ 3 keys

K_1 K_2 K_3 → K_3 K_2 K_1

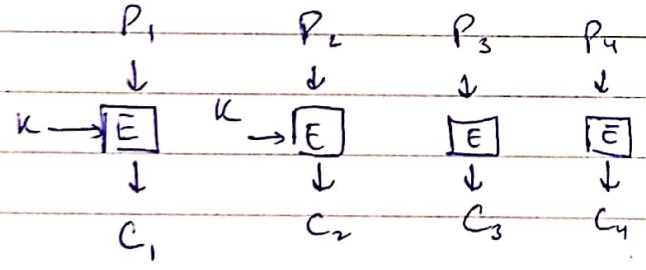
- unable to exploit as in E last key is key 3 which is different then K1 unlike in Triple E/D w/ 2 keys.
- backward compatible w/ Triple E/D with 2 keys.

Modes of Operation :

- how to handle different blocks / algo to adapt for an application
- 5 modes

(1) Electronic Codebook:

- DES, AES etc any code block oriented algo
- treat each block independently using same key
- 256 bits — 64 64 64 64 — 4 blocks (P_1 P_2 P_3 P_4)



- pattern found • if repitition in P.
- random access (can access any part for E/D).

- secure transmission of single values. e.g. encryption key.

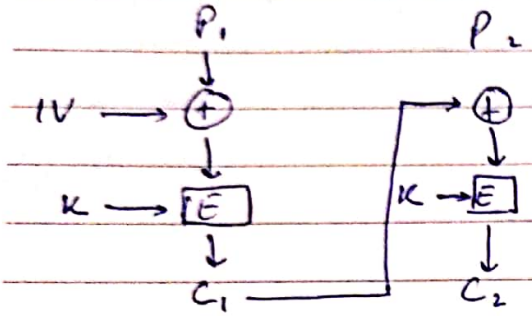
Evaluation Criteria
- if one bit changes, check and fix only that block — independent
- errored bit doesn't effect other P blocks — no propagation
- less overhead
- diffusion : discarding this technique on basis of diffusion.

* can't make channel secure so make info secure

Date

(2) Cipher Block Chaining:

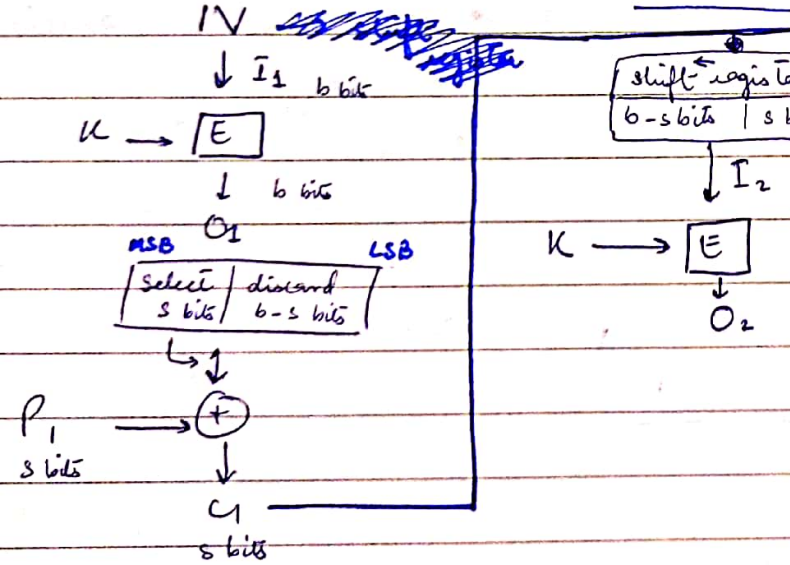
- input to E algo is: Next block of P ⊕ preceding block C.
- used for authentication, general purpose transmission
- P ⊕ Initial Vector ← randomly generated
  communicated to D side securely
  size = block size 128 b if AES
- IV being used once.



(3) Cipher feedback mode:

- conversion of block to stream.
- 3 modes: cipher feedback mode, output feedback mode, counter mode

∴ block size = s-bits initial vector: b bits b>s



← shift s bits and in free space put s bits of C_1 (not circular shift)

- in this algo, IV is being transmitted in C_i and in stream unlike in CBC

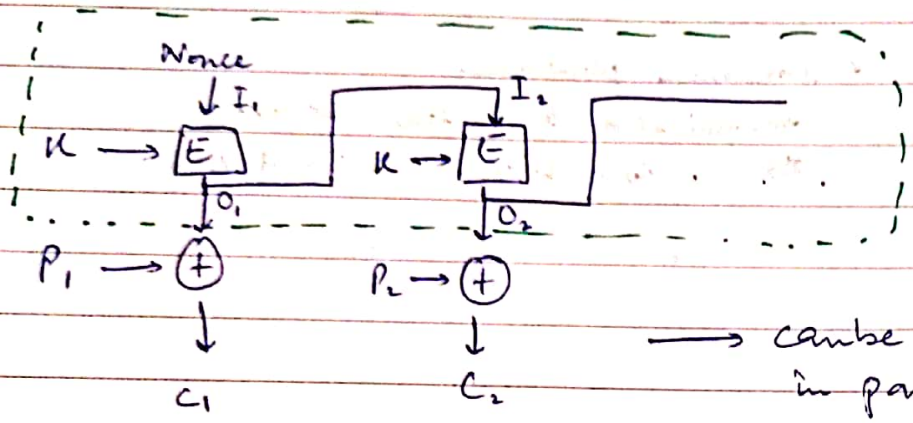
CONV: no independency.

Nonce = no. used only once

Date

(4) Output Feedback Mode:

- no issue of dependancy

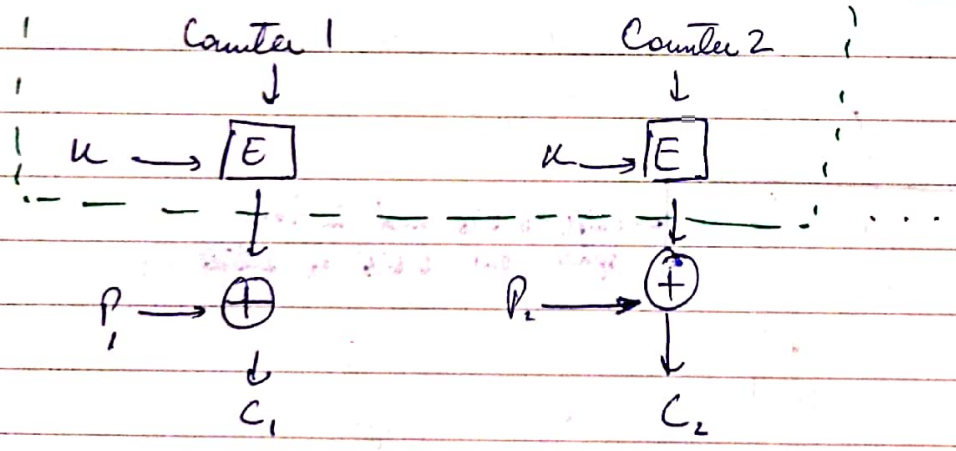


- security ↓
- makes it quicker

→ preprocessed before generating ciphers.
we have K and Nonce
↓
required once

→ can be done in parallel

(5) Counter Mode:



- counter 1 be fixed, counter 2 can be C1 + 1...
- can run in parallel
- HW efficiency
- random access → if you want to access (E/D) any block in between then can do that
- SW efficiency

Chapter 9 Public Key Encryption + RSA

Asymmetric keys: two related keys, public (known to public) + private (confidential)

Misconceptions about Asym. Enc.: public key can be vulnerable,

Principles of Public-Key Cryptosystems:

- Key distribution
  - secure comm. without have to trust Key Dist. Center with your key
- Digital signatures
  - verify if msg came from original sender

Encryption with public key: in slides

intend: msg secure + secure info

$$Y = E(PU_a, X)$$

Y: output, X: input

Encryption with Private Key: in slides

intend: digital sign

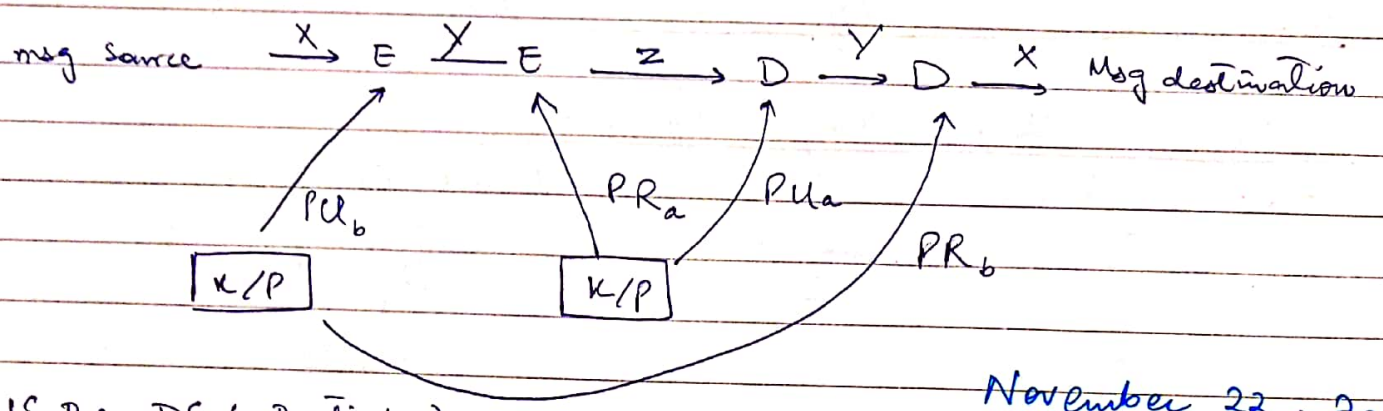
$$Y = E(PR_b, X)$$

digital signature: only the relevant/original authority can use their private key. Bob decrypts the msg with his own private key. so, it's understood msg came from Bob for sure.

* do this when you want to tell it is from sent by me for sure because public K is with anyone. anyone can decrypt + read msg.

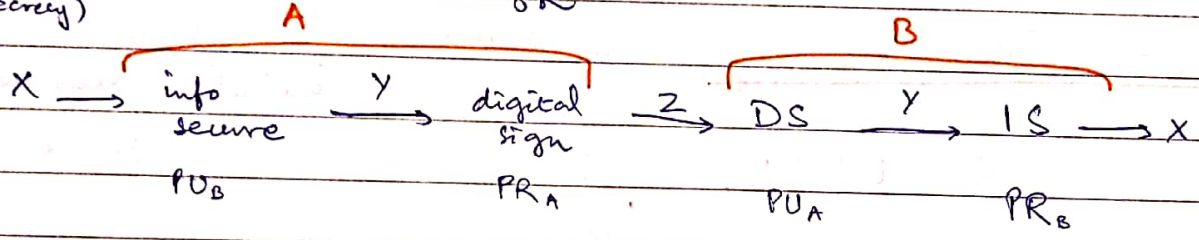
Table 9.2. Difference b/w Symmetric / Asymmetric

- key pair source generates key pairs — external body
  B party generates key pairs — internal op.



November 22, 2021

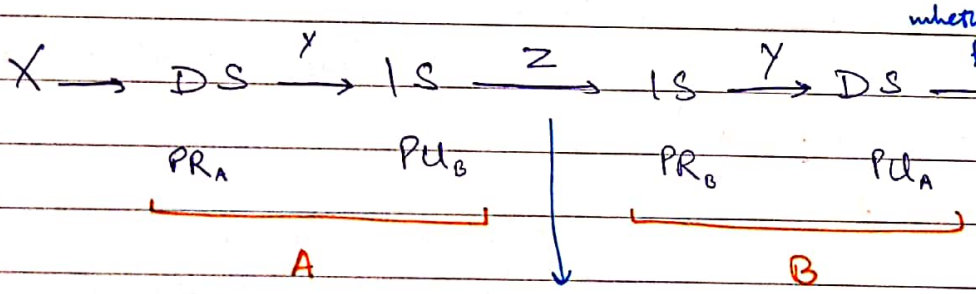
first IS then DS (authentication) OR (secrecy)



↳ problem in IS → DS: when ~~B~~ gets something in DS, B doesn't know whether the 'something' is really from A or not. B must do IS to see. If they get meaningful msg then we know along this is from A.

(We assume that recipient already knows from whom they will recieve a ~~to~~ msg

first auth. then secrecy DS → IS



(PU_B, PR_B)

There should be no pattern/relation in them.

Slide 11