Unit 2: Block Ciphers and Advanced Encryption Standard

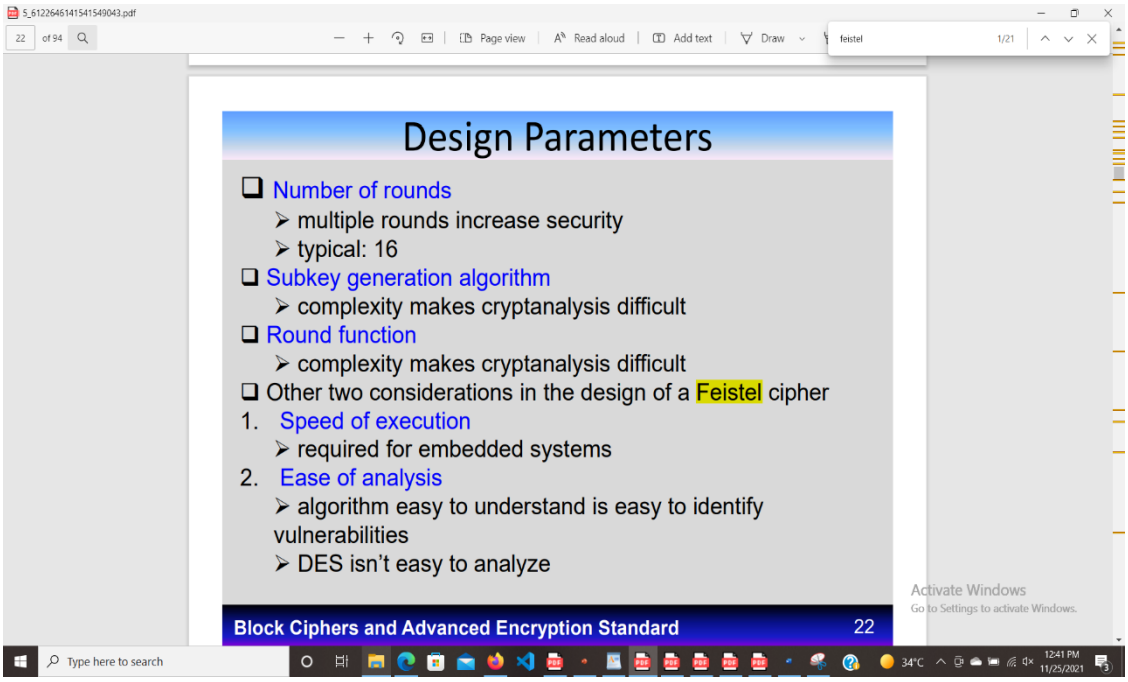
1) What are the parameters and design features for realization of a Feistel network.

ANS:

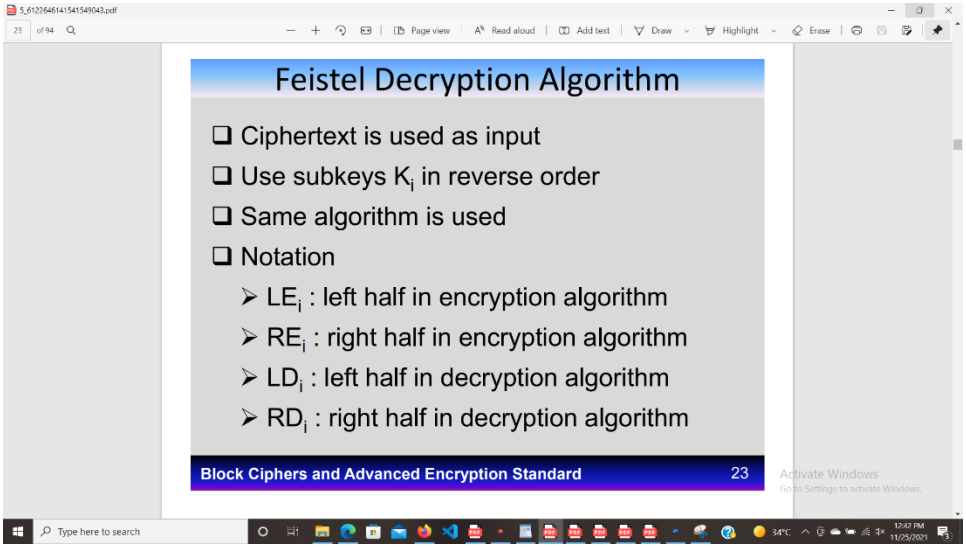
The exact realization of a Feistel network depends on the choice of the following parameters and design features:

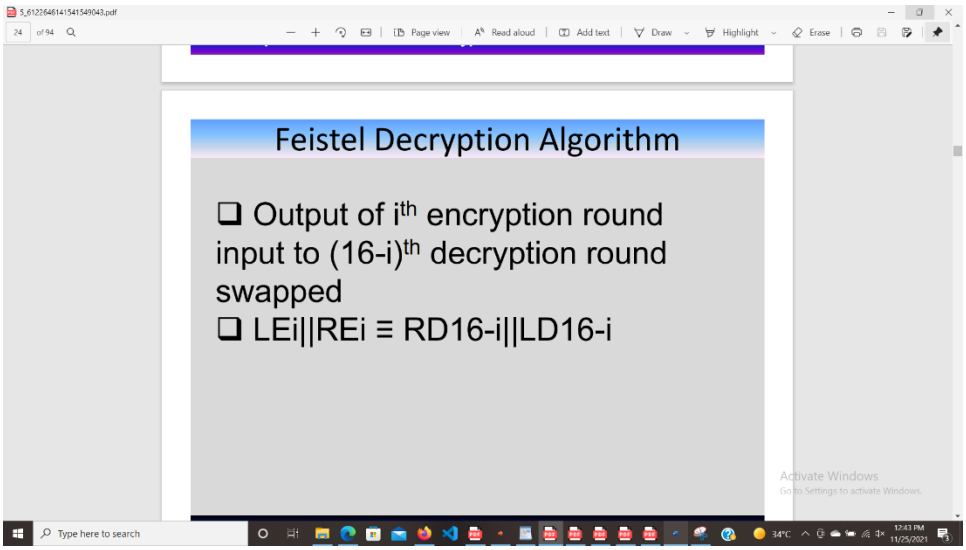
Design Parameters

- ❑ Block size
  - ➢ larger: greater security (diffusion)
  - ➢ smaller: faster encryption, decryption
  - ➢ typical: 64 bit, 128 bit AES
- ❑ Key size
  - ➢ larger: greater security (brute-force resist)
  - ➢ smaller: faster encryption, decryption
  - ➢ typical: 128 bit

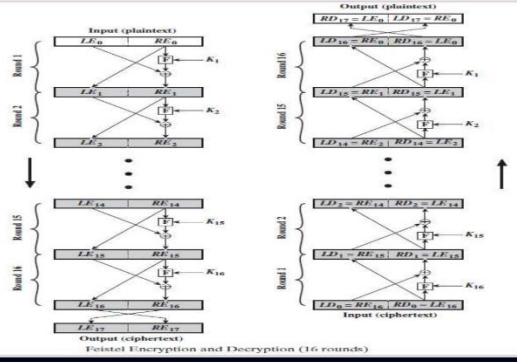


2) Explain Feistel decryption algorithm.





Feistel Decryption Algorithm



Feistel Encryption and Decryption (16 rounds)

Decryption Proof

- Encryption side
  - LE16 = RE15
  - RE16 = LE15 ⊕ F(RE15, K16)
- Decryption side
  - LD1 = RD0 = LE16 = RE15
  - LD0 = RE16
  - RD1 = LD0 ⊕ F(RD0, K16)
    = RE16 ⊕ F(RE15, K16)
    = [LE15 ⊕ F(RE15, K16)] ⊕ F(RE15, K16)
- XOR properties
  - [A ⊕ B] ⊕ C = A ⊕ [B ⊕ C]
  - D ⊕ D = 0 **AND** E ⊕ 0 = E
- Thus
  RD1 = [LE15 ⊕ F(RE15, K16)] ⊕ F(RE15, K16)
  = **LE15**
- **Similarly LD1 = RE15**

Decryption Proof

- This correspondence holds all the way through the 16 iterations, as is easily shown
- In general terms.

$LE_i = RE_{i-1}$

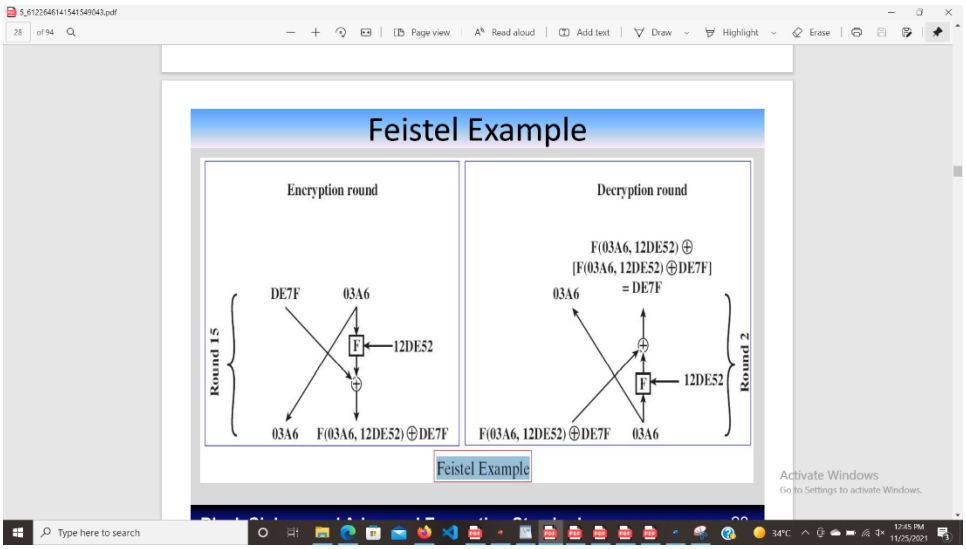
$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$

Rearranging terms:

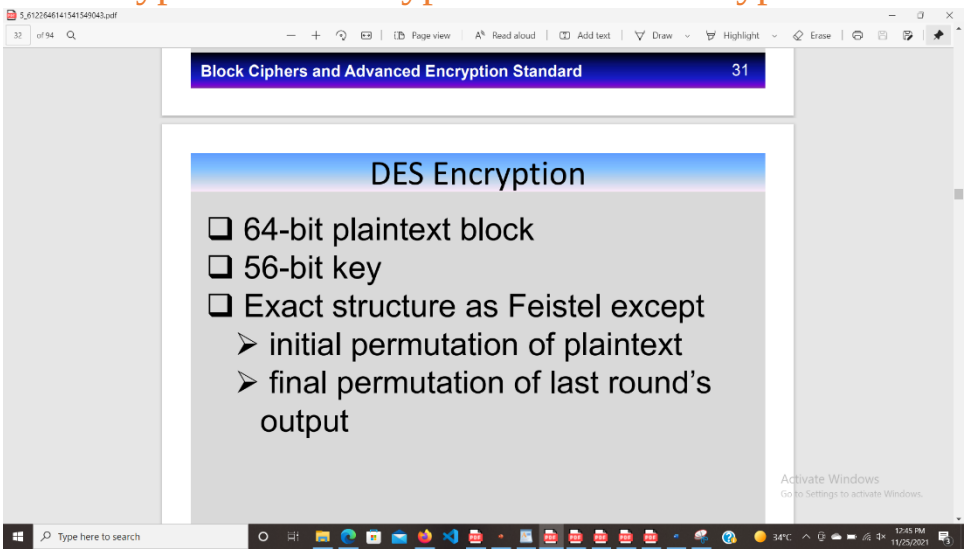
$RE_{i-1} = LE_i$

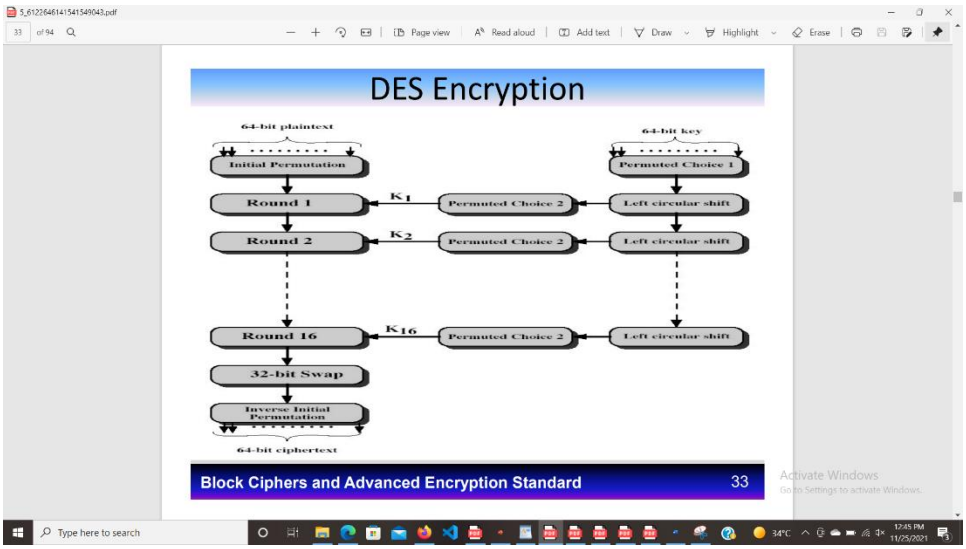
$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i)$

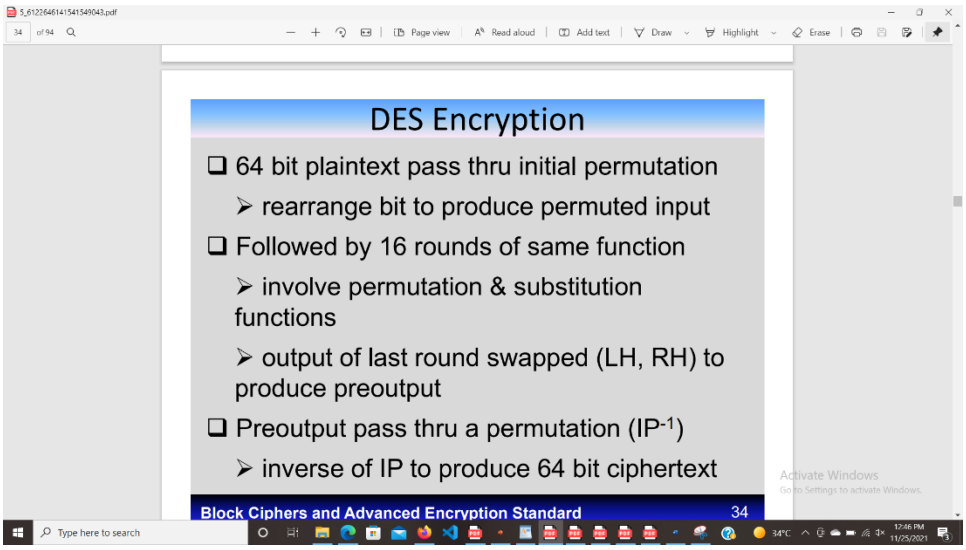
- Output of the last round of the decryption process is $RE_0 \,||\, LE_0$

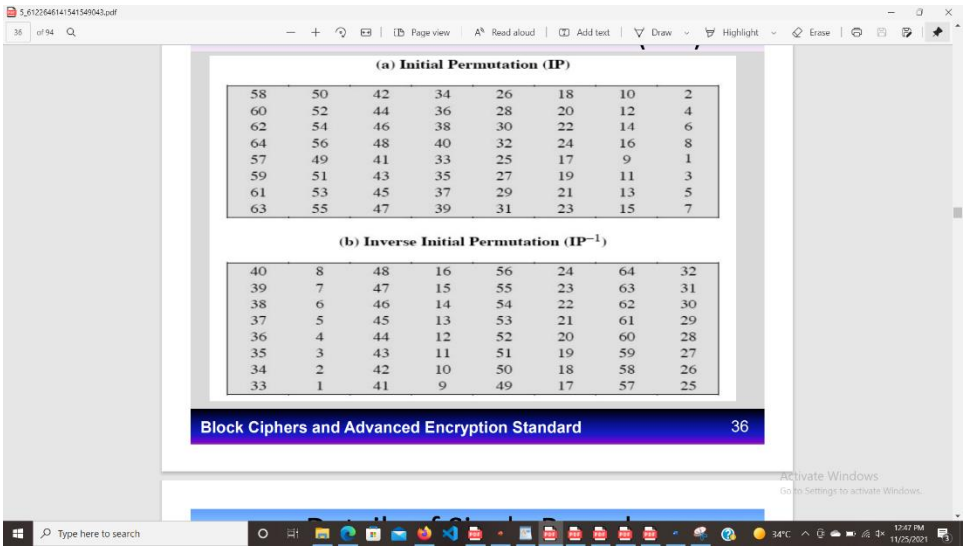


3) Explain encryption and decryption in Data Encryption Standard (DES).









(a) Initial Permutation (IP)

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

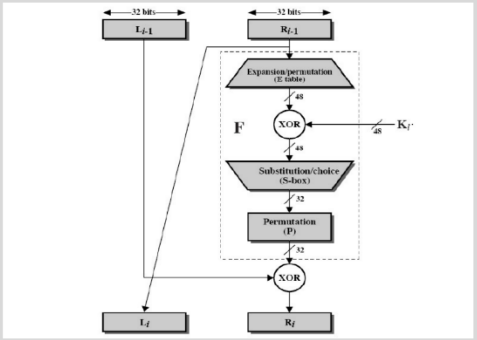
(b) Inverse Initial Permutation (IP^{-1})

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

Details of Single Round

- Left and right halves of 64 bits are separated into two 32-bit parts L,R
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1},K_i)$
- R is expanded to 48 bits using permutation E
- Resulting 48 bits are XORed with K_i
- 48 bit result passes thru substitution function F (8 S-boxes) producing 32-bit output
- Output is permuted using permutation function P

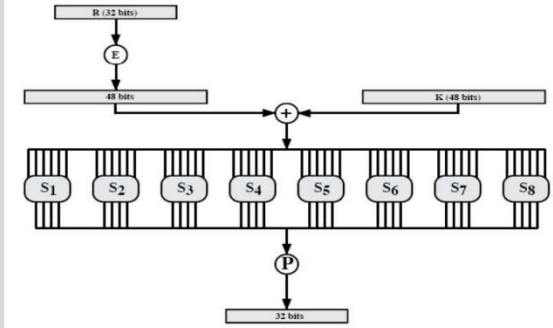
Details of Single Round



Role of S-Boxes

- 8 s-boxes, each has 6 bits input, 4 bits out
- outer 2 bits (1,6) used to select row
- inner 4 bits (2-5) used to select column
- decimal value of cell converted to 4 bits out
  - note that decimal values are [0-15]
- 8 4-bit groups produce 32 bit output

Role of S-Boxes



Permutation Tables E, P

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

Definition of DES S-Boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Definition of DES S-Boxes

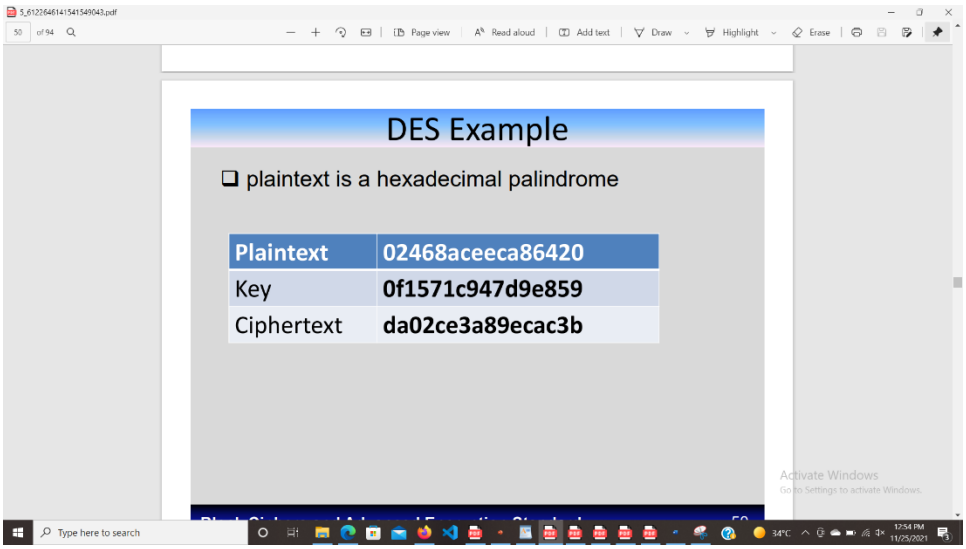
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

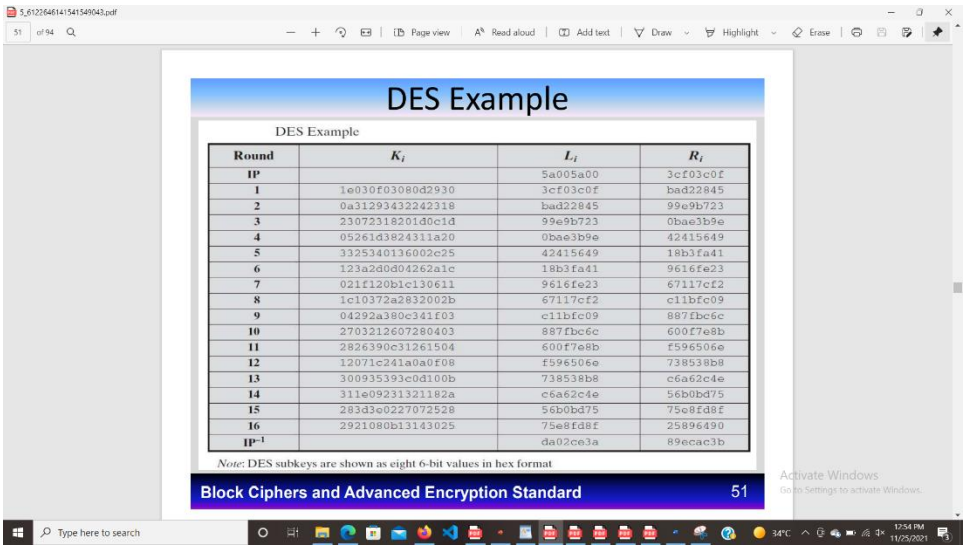
Example

- Using S1
- Input: 011001
- Row is 01: (1)
- Column 1100: (12)
- Value of row 1, column 12 is 9
- Output is 1001

DES Decryption

- As with any Feistel cipher, decryption uses the same algorithm as encryption,
- subkeys are reversed.



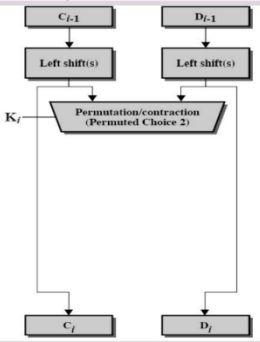


4) Explain key generation in Data Encryption Standard (DES).

Key Generation

- 64-bit key used as input (8 × 8 table)
- 8th bit in each row is ignored →56 bits
- key is permuted using table PC-1
- resulting 56 bits separated into two 28-bit parts C0, D0
- Each round
  - circular left shift C_{i-1}, D_{i-1} of 1 or 2 bits (table)
  - shifted values go to next round
  - also used as input to table PC-2
  - PC-2 produce 48-bit output Ki used in $F(R_{i-1}, K_i)$

Key Generation



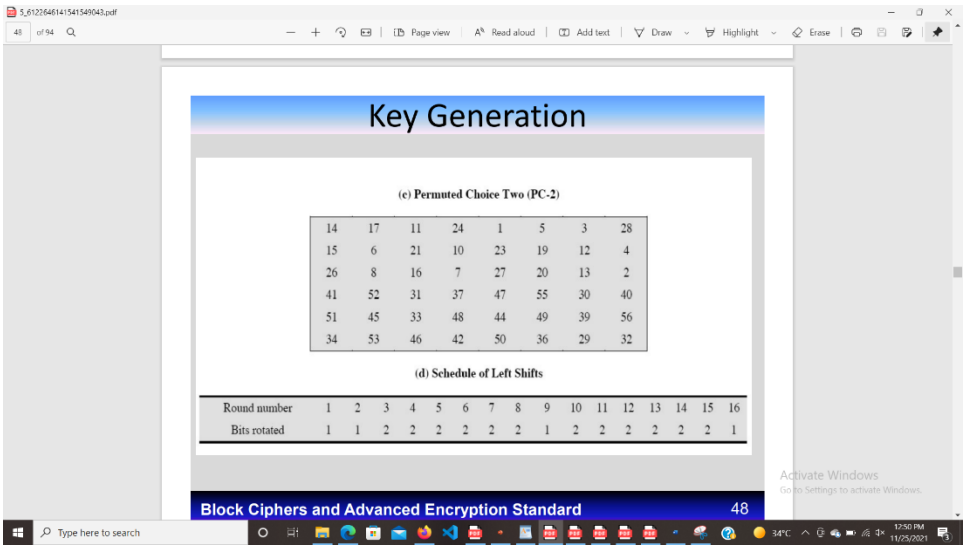
Key Generation

(a) Input Key

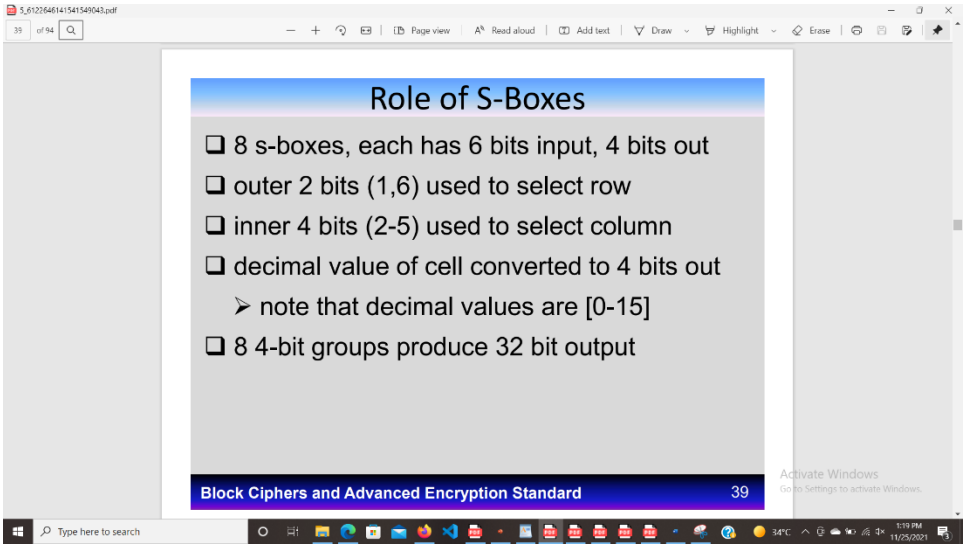
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

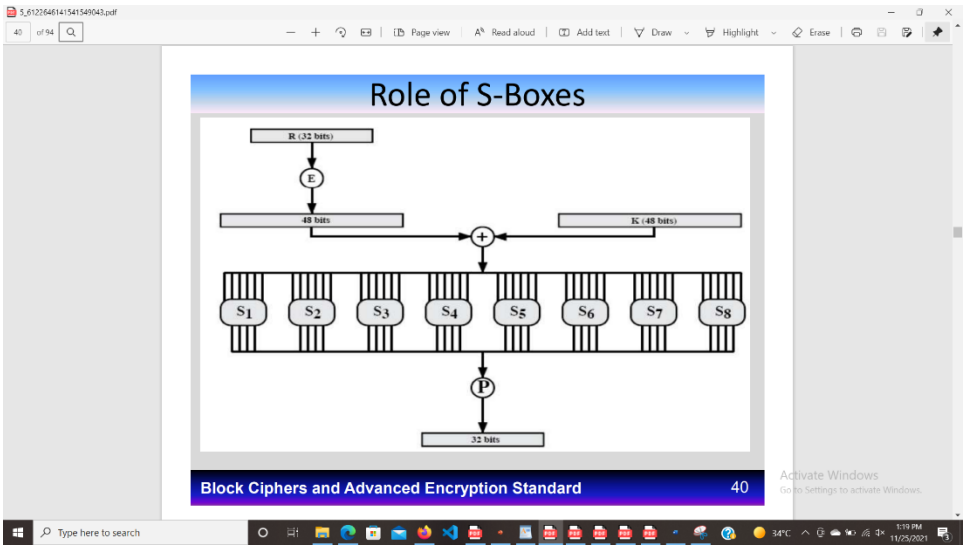
(b) Permuted Choice One (PC-1)

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

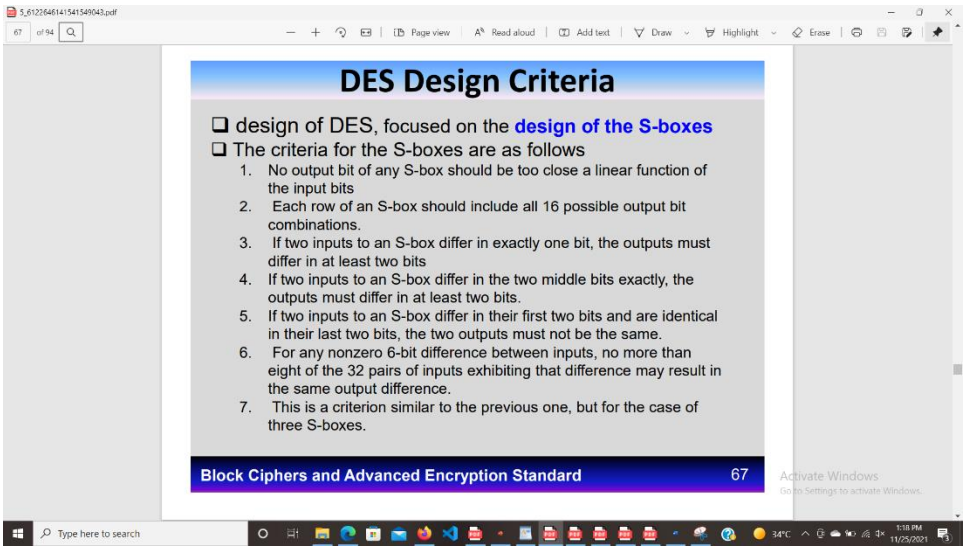


5) What is the purpose of the S-boxes in Data Encryption Standard (DES)?

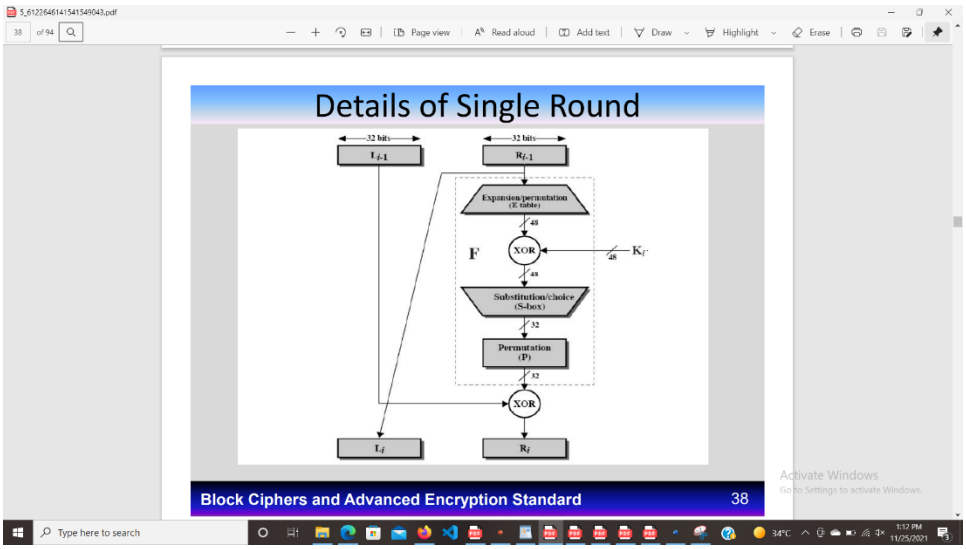


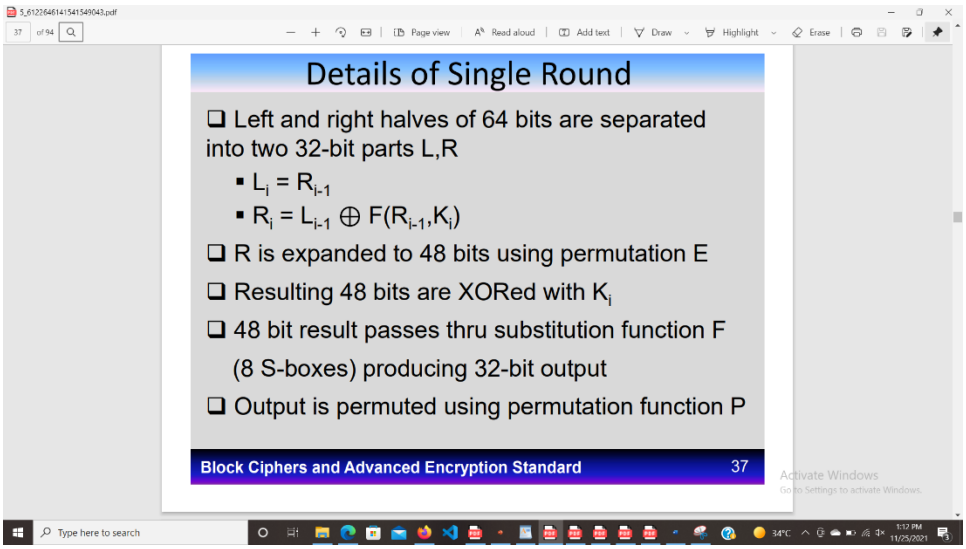


6) Explain operation of S-Boxes in Data Encryption Standard (DES).

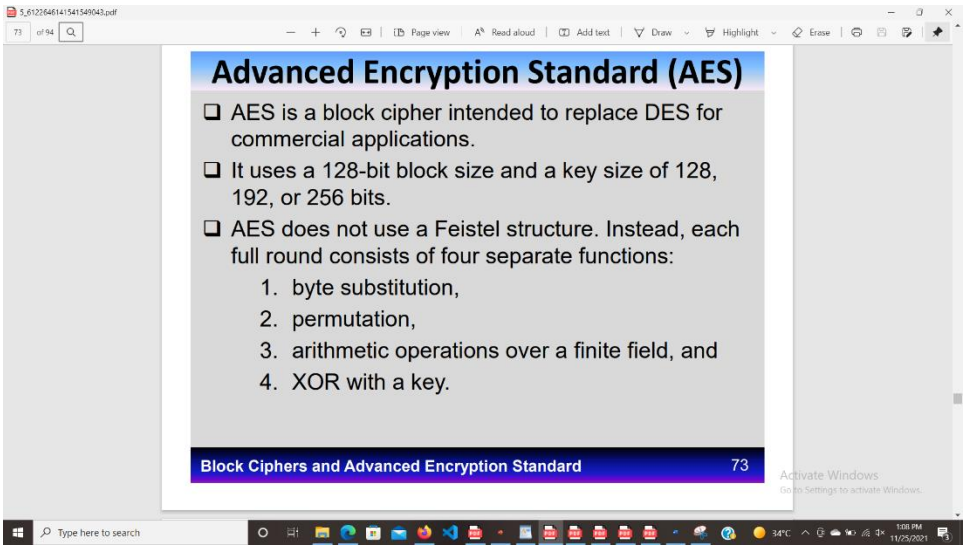


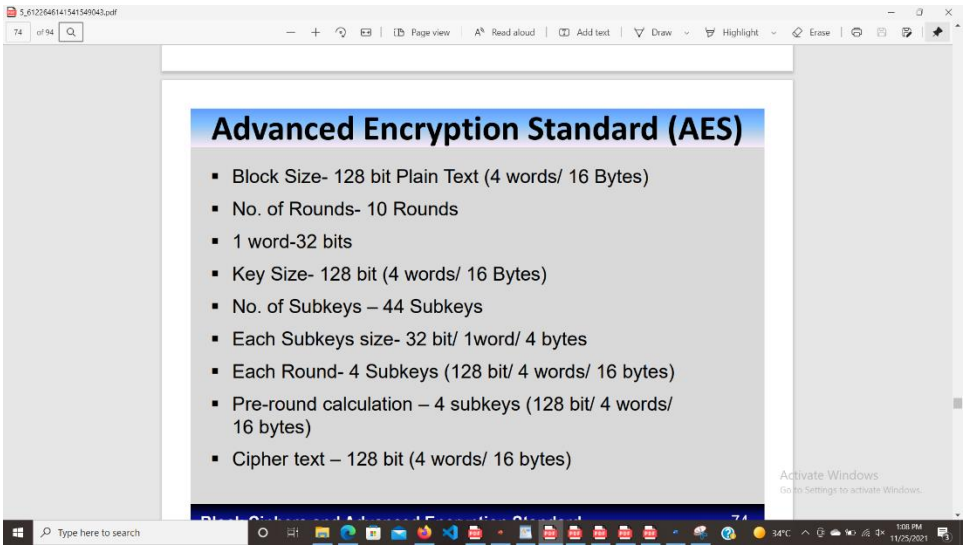
7) Explain Single Round of DES Algorithm with neat diagram.

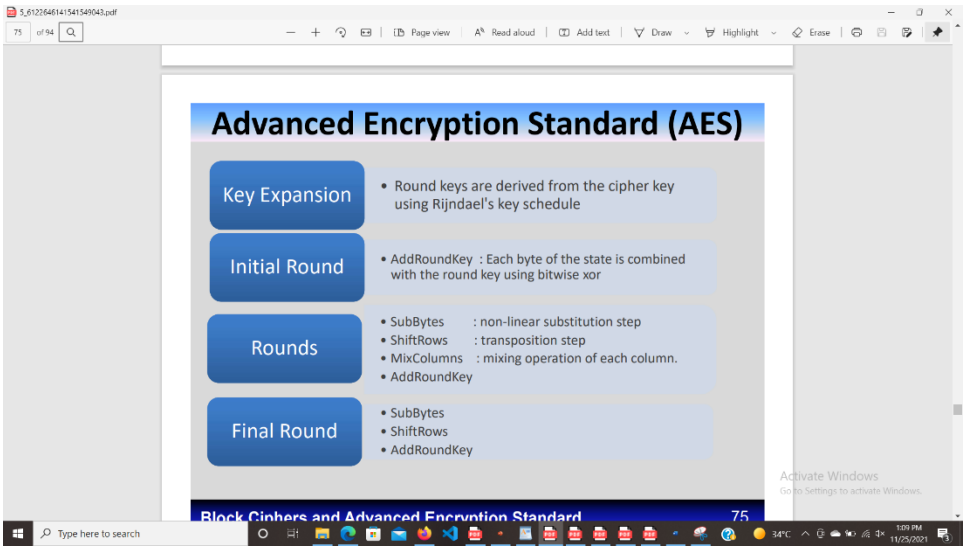


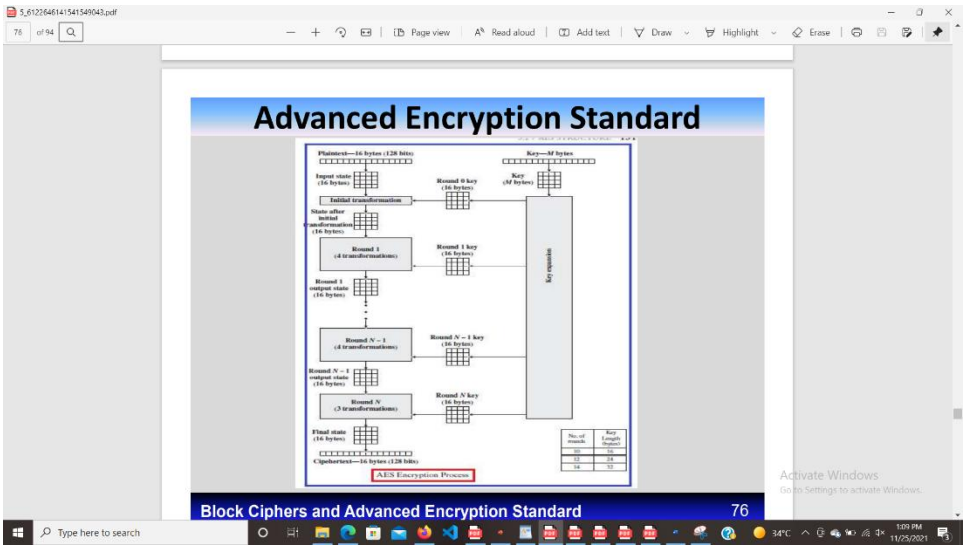


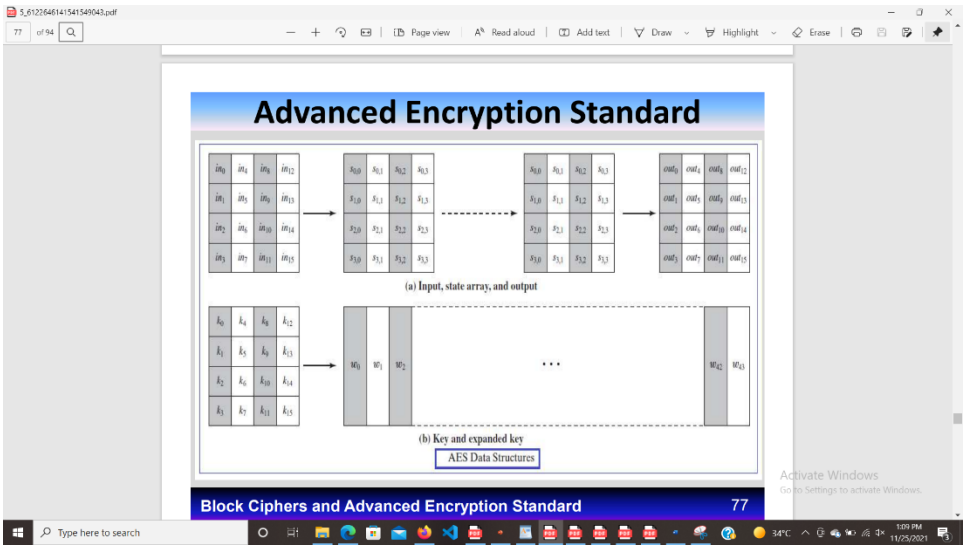
8) Explain general structure of Advanced Encryption Standard (AES).


Advanced Encryption Standard (AES)
AES is a block cipher intended to replace DES for commercial applications.
It uses a 128-bit block size and a key size of 128, 192, or 256 bits.
AES does not use a Feistel structure. Instead, each full round consists of four separate functions:
1. byte substitution,
2. permutation,
3. arithmetic operations over a finite field, and
4. XOR with a key.
Block Ciphers and Advanced Encryption Standard 73


Advanced Encryption Standard (AES)
Block Size- 128 bit Plain Text (4 words/ 16 Bytes)
No. of Rounds- 10 Rounds
1 word-32 bits
Key Size- 128 bit (4 words/ 16 Bytes)
No. of Subkeys – 44 Subkeys
Each Subkeys size- 32 bit/ 1word/ 4 bytes
Each Round- 4 Subkeys (128 bit/ 4 words/ 16 bytes)
Pre-round calculation – 4 subkeys (128 bit/ 4 words/ 16 bytes)
Cipher text – 128 bit (4 words/ 16 bytes)

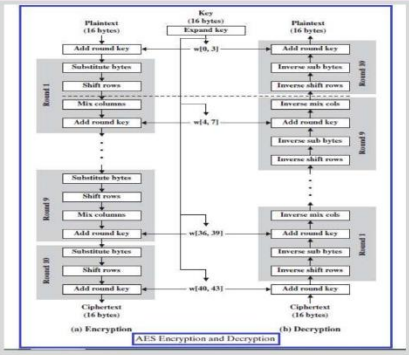

Advanced Encryption Standard (AES)
Key Expansion
Round keys are derived from the cipher key using Rijndael's key schedule
Initial Round
AddRoundKey : Each byte of the state is combined with the round key using bitwise xor
Rounds
SubBytes : non-linear substitution step
ShiftRows : transposition step
MixColumns : mixing operation of each column.
AddRoundKey
Final Round
SubBytes
ShiftRows
AddRoundKey
Block Ciphers and Advanced Encryption Standard 75





9) Explain detailed structure of Advanced Encryption Standard (AES).

AES Detailed Structure



AES Detailed Structure

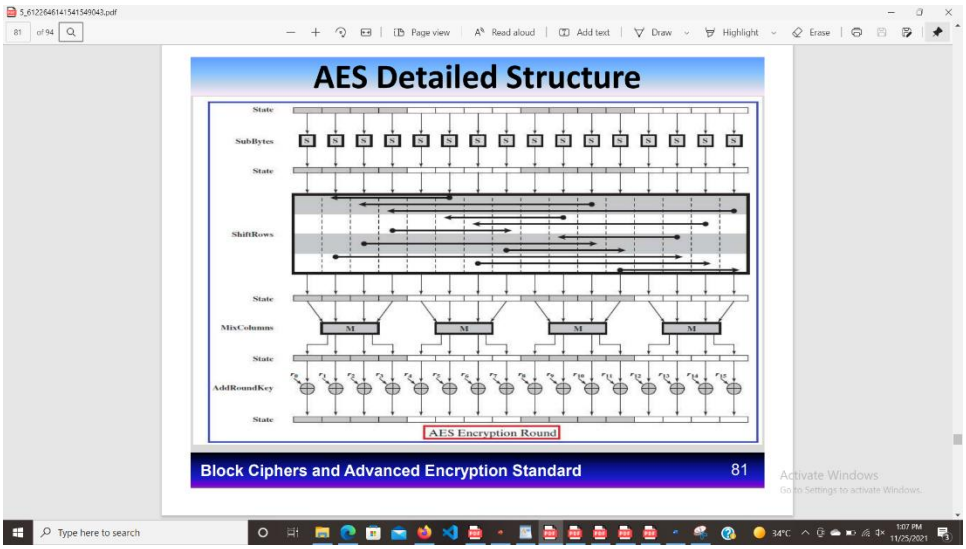
Overall AES structure:

- One noteworthy feature of this structure is that it is not a Feistel structure.
- The key that is provided as input is expanded into an array of forty-four 32-bitwords, **w**[*i*]. Four distinct words (128 bits) serve as a round key for each round
- Four different stages are used, one of permutation and three of substitution
  1. **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block
  2. **ShiftRows:** A simple permutation
  3. **MixColumns:** A substitution that makes use of arithmetic over GF(2^8)
  4. **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key
- The structure is quite simple. For both encryption and decryption, the

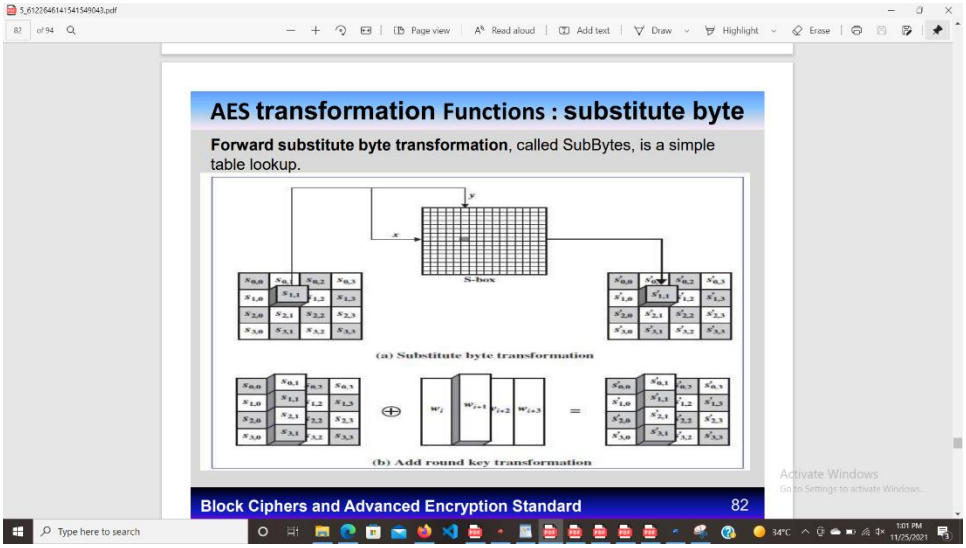
AES Detailed Structure

Overall AES structure:

- Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage.
- The AddRoundKey stage is, in effect, a form of Vernam cipher
- Each stage is easily reversible.
- As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order.
- Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext.
- The final round of both encryption and decryption consists of only three stages.



10)Explain Advanced Encryption Standard (AES) transformation functions.



AES transformation Functions : substitute byte

AES S-Boxes

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

(a) S-box

AES transformation Functions : substitute byte

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

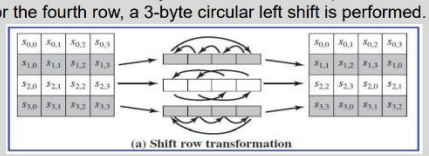
(b) Inverse S-box

AES transformation Functions : substitute byte

- The **inverse substitute byte transformation**, called InvSubBytes, makes use of the inverse S-box
- Note, for example, that the input {2A} produces the output {95}, and the input {95} to the Inverse S-box produces {2A}.

AES transformation Functions: ShiftRows

- The **forward shift row transformation**, called ShiftRows.
- The first row of **State** is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.

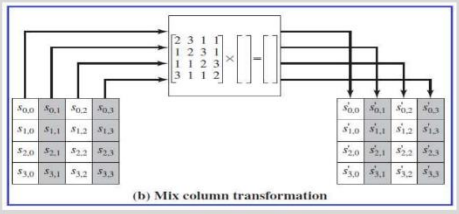


(a) Shift row transformation

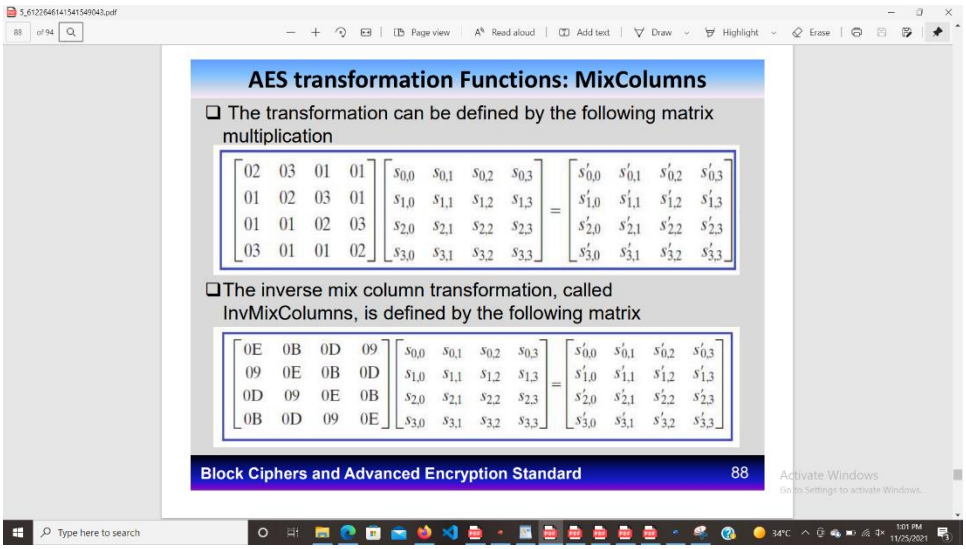
87	F2	4D	97	→	87	F2	4D	97
EC	6E	4C	90		6E	4C	90	EC
4A	C3	46	E7		46	E7	4A	C3
8C	D8	95	A6		A6	8C	D8	95

AES transformation Functions: MixColumns

- The **forward mix column transformation**, called MixColumns, operates on each column individually.
- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.



(b) Mix column transformation



AES transformation Functions: MixColumns

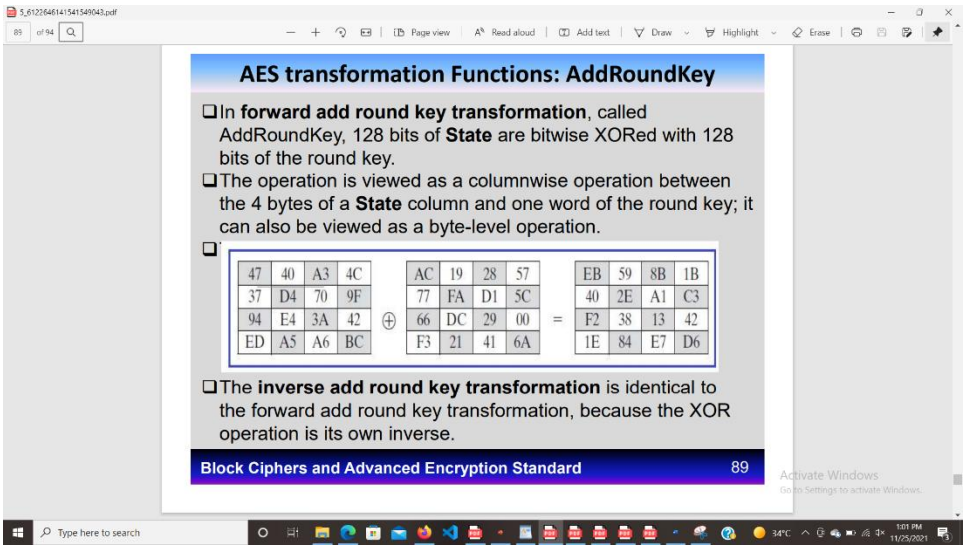
- The transformation can be defined by the following matrix multiplication

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- The inverse mix column transformation, called InvMixColumns, is defined by the following matrix

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Block Ciphers and Advanced Encryption Standard 88



AES transformation Functions: AddRoundKey

- In **forward add round key transformation**, called AddRoundKey, 128 bits of **State** are bitwise XORed with 128 bits of the round key.
- The operation is viewed as a columnwise operation between the 4 bytes of a **State** column and one word of the round key; it can also be viewed as a byte-level operation.
-

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

(⊕ between the first two matrices, = before the third)

- The **inverse add round key transformation** is identical to the forward add round key transformation, because the XOR operation is its own inverse.

Block Ciphers and Advanced Encryption Standard 89

11) Explain Advanced Encryption Standard (AES) key expansion.

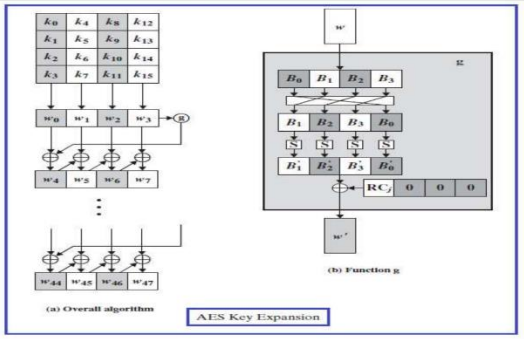
AES KEY EXPANSION

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

```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i], key[4*i+1],
                                       key[4*i+2],
                                       key[4*i+3]);

    for (i = 4; i < 44; i++)
    {
     temp = w[i - 1];
     if (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                              ⊕ Rcon[i/4];
     w[i] = w[i-4] ⊕ temp
    }
}
```


AES KEY EXPANSION



Implementation Aspects

- The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.
- Very efficient
- Implementation was a key factor in its selection as the AES cipher
- AES animation: