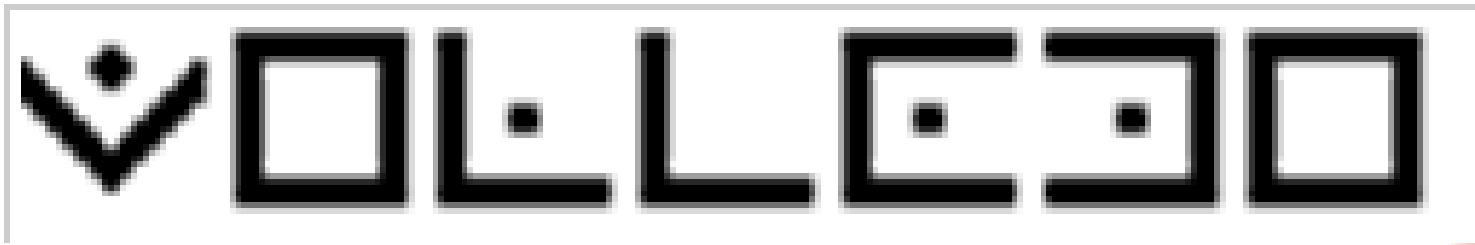
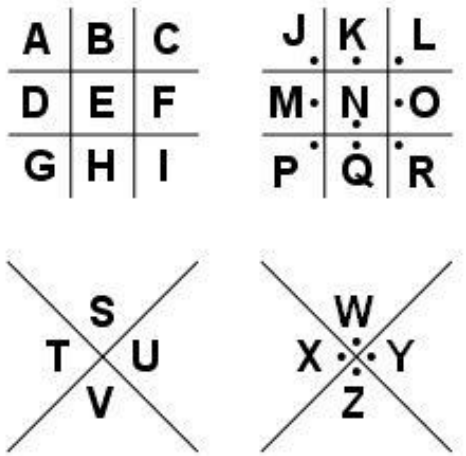
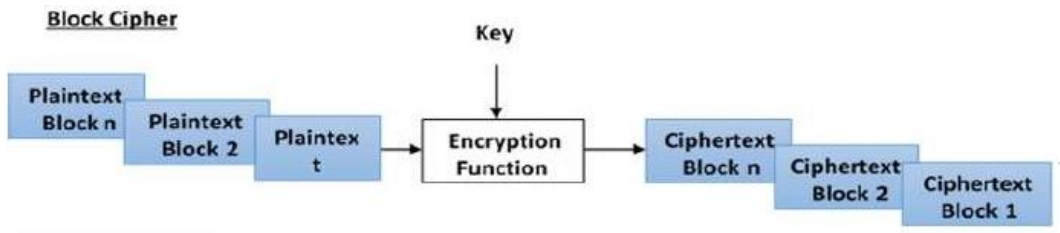
MODERN BLOCK CIPHERS
DES & AES







Mukesh Chinta

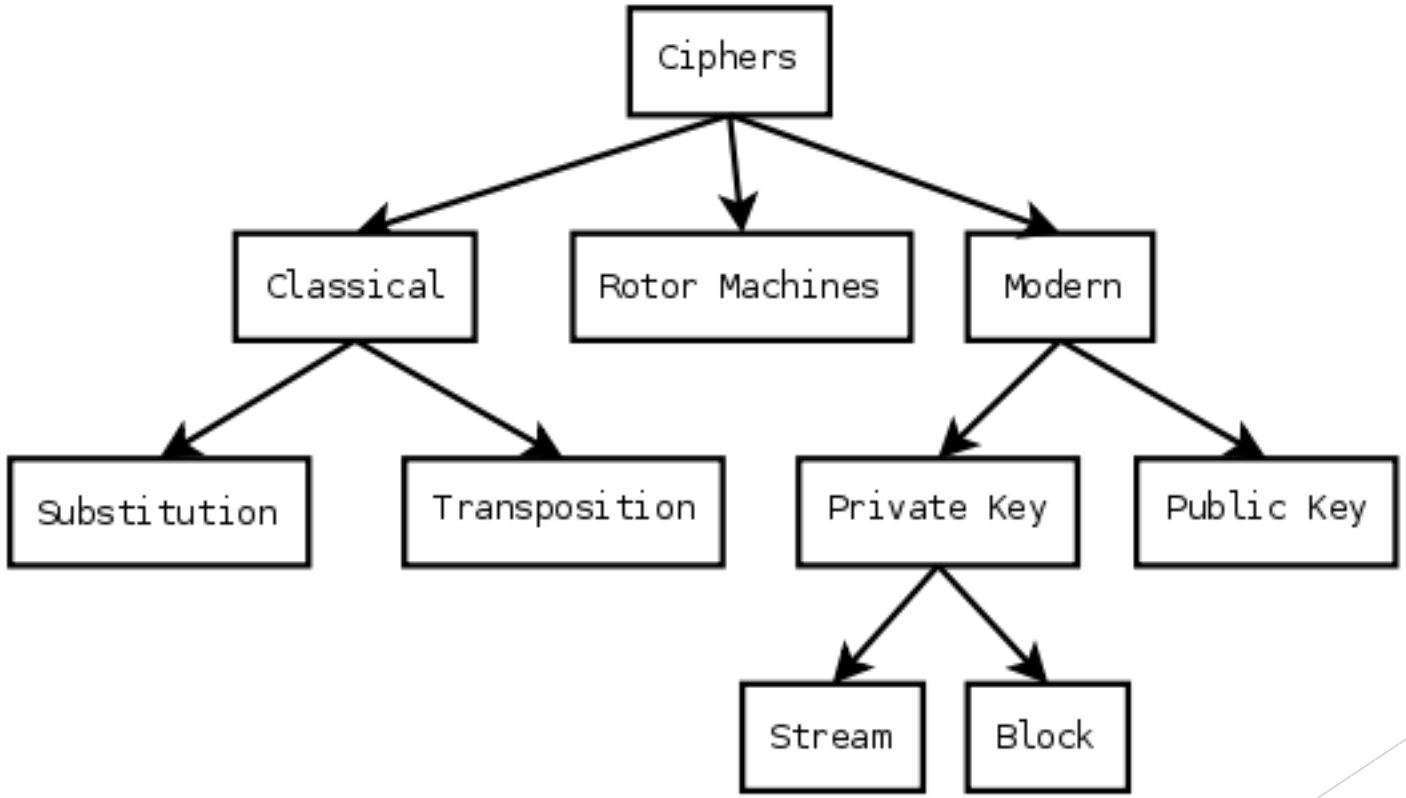
Assistant Professor

Dept of CSE

VRSEC

Ciphers

- In cryptography, a cipher is an algorithm for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure. An alternative, less common term is Encipherment.



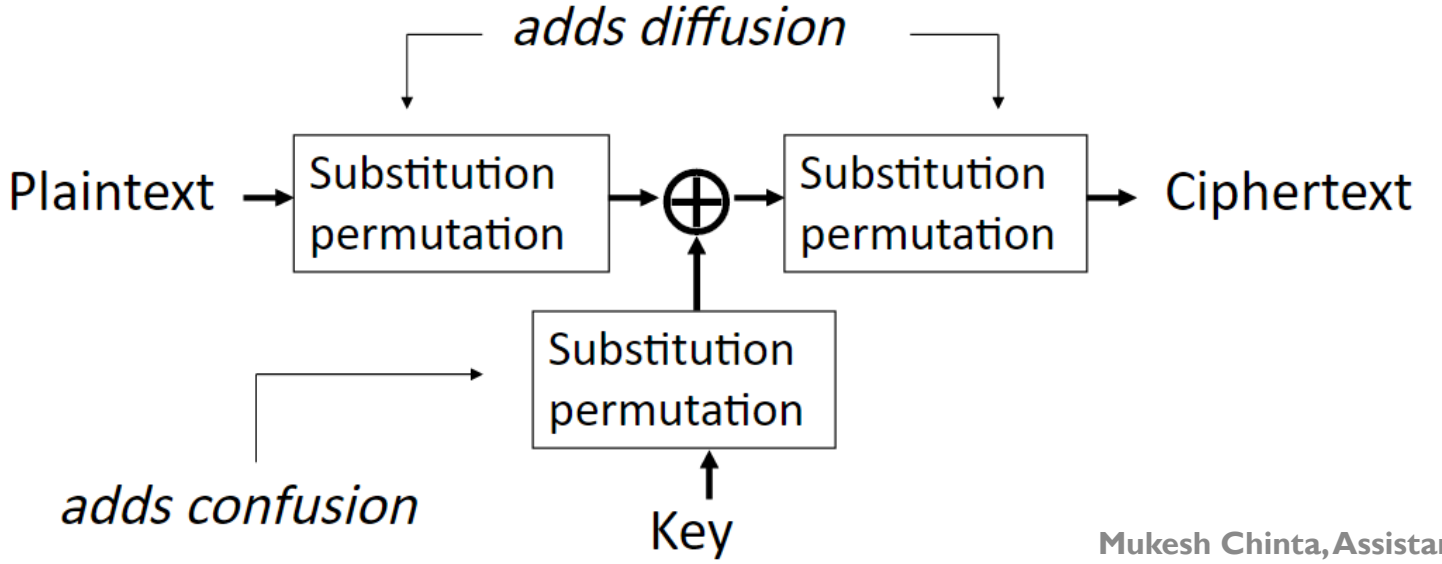
Comparison of Block & Stream Ciphers

S.N.	Stream cipher	Block cipher
1.	Stream ciphers operate on smaller units of plaintext.	Block ciphers operate on larger block of data.
2.	Faster than block cipher	Slower than stream cipher
3.	Stream cipher processes the input element continuously producing output one element at a time.	Block cipher processes the input one block of element at a time, producing an output block for each input block.
4.	Requires less code	Requires more code
5.	Only one time of key use.	Reuse of key is possible
6.	Ex. - One time pad	Ex. - DES
7.	Application - SSL (secure connections on the web.)	Application - Database, file encryption.
8.	Stream cipher is more suitable for hardware implementation	Easier to implement in software.

Modern Block Ciphers

- Modern block ciphers are widely used to provide encryption of quantities of information, and/or a cryptographic checksum to ensure the contents have not been altered.
- Block ciphers work a on block / word at a time, which is some number of bits. All of these bits have to be available before the block can be processed. Stream ciphers work on a bit or byte of the message at a time, hence process it as a “stream”. Block ciphers are currently better analysed, and seem to have a broader range of applications.
- Most symmetric block encryption algorithms in current use are based on a structure referred to as a **Feistel block cipher**. A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits.
- Claude Shannon’s 1949 paper has the key ideas that led to the development of modern block ciphers. Critically, it was the technique of layering groups of S-boxes separated by a larger P-box to form the S-P network, a complex form of a product cipher. He also introduced the ideas of *confusion* and *diffusion*, notionally provided by S-boxes and P-boxes

- Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key.
- The mechanism of **diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce the key.
- **Confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.



Feistel Cipher Structure

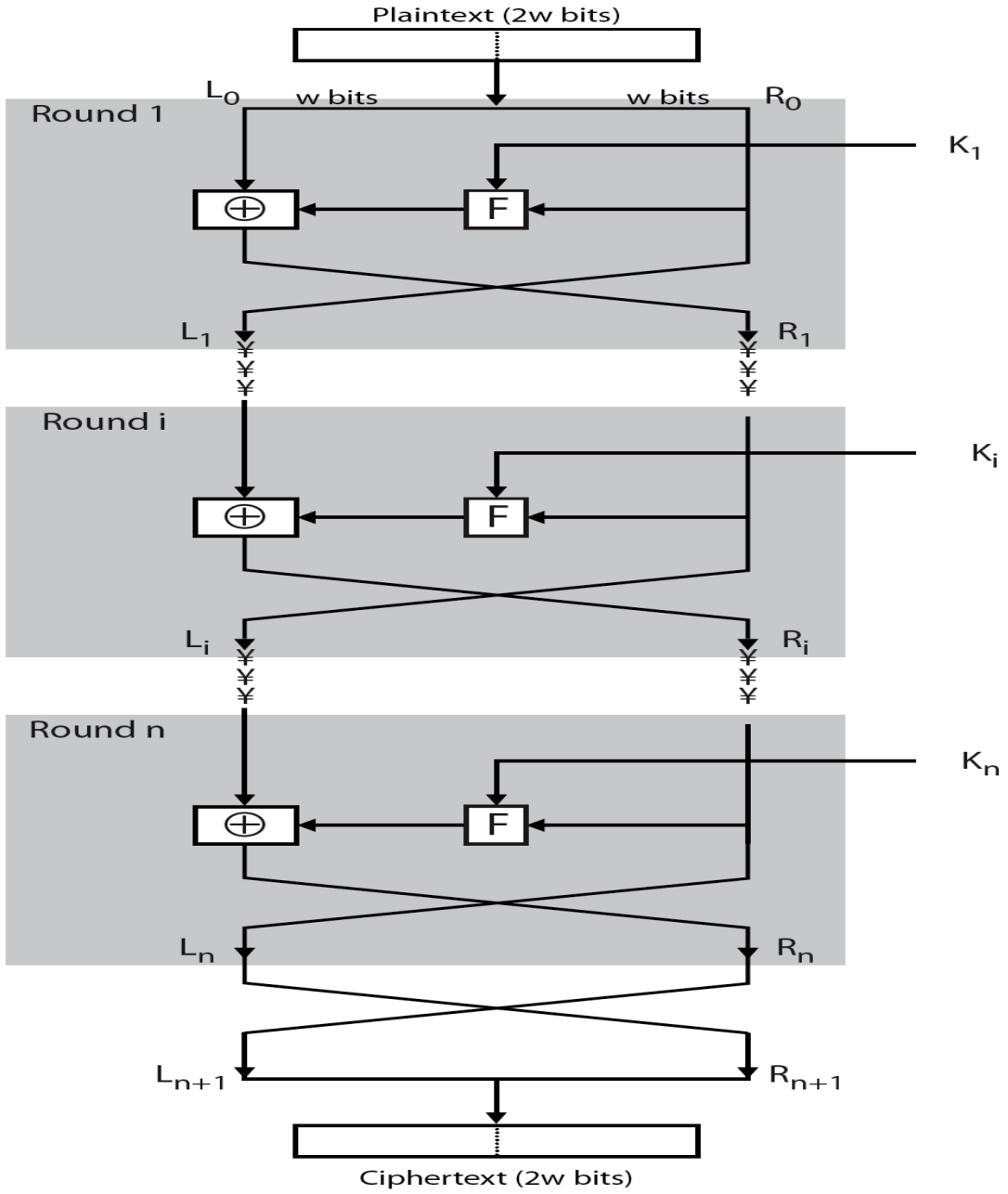
- Horst Feistel, working at IBM Thomas J Watson Research Labs devised a suitable invertible cipher structure in early 70's. First described by Fiestel in 1973, it forms the basis for almost all conventional encryption schemes

- It is depends on the choice of the following parameters
  - **Block size:** larger block sizes mean greater security (typical size is 64 bits)
  - **Key Size:** larger key size means greater security (typical size is 128 bits)
  - **Number of rounds:** multiple rounds offer increasing security but slows down the cipher (N = 16)
  - **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
  - **Round Function (F):** Greater complexity in this algorithm means greater resistance
  - **Fast software encryption/decryption:** the speed of execution of the algorithm becomes a concern.
  - **Ease of Analysis:** Though the algorithm is made as difficult as possible to cryptanalyze, it should be easy to analyze.

One of Feistel's main contributions was the invention of a suitable structure which adapted Shannon's S-P network in an **easily inverted structure**. It partitions input block into two halves which are processed through multiple rounds which perform a substitution on left data half, based on round function of right half & subkey, and then have permutation swapping halves. Essentially the same h/w or s/w is used for both encryption and decryption, with just a slight change in how the keys are used.

Steps:

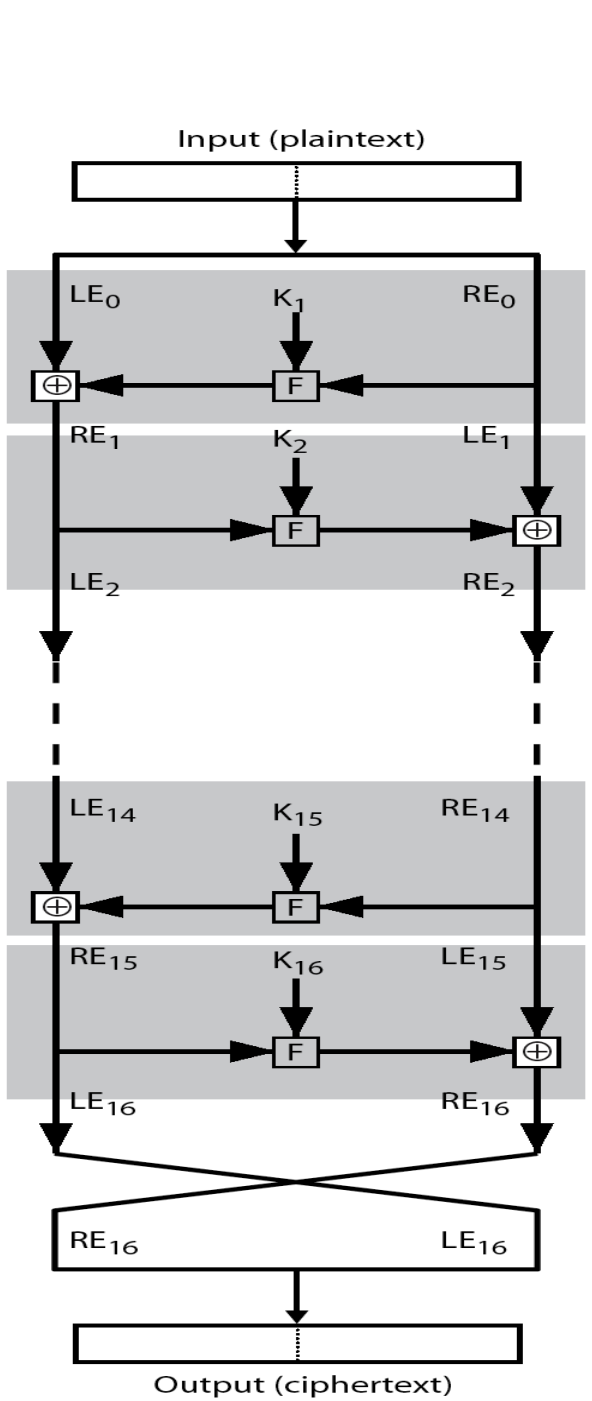
- Input of plaintext with length 2w bits and key K.
- Plaintext is divided into two halves L**0** and R**0**.
- These two halves pass through N rounds of processing to produce CipherText block.
- The key K is derived from subkey generation algorithm.
- These two halves combine by applying a round function '**F**' on right half of data and then taking XOR operation of the output of **F** with left half of data.
- A permutation is performed which swaps the two halves of data forming a substitution – permutation network (SPN) as proposed by Shannon.

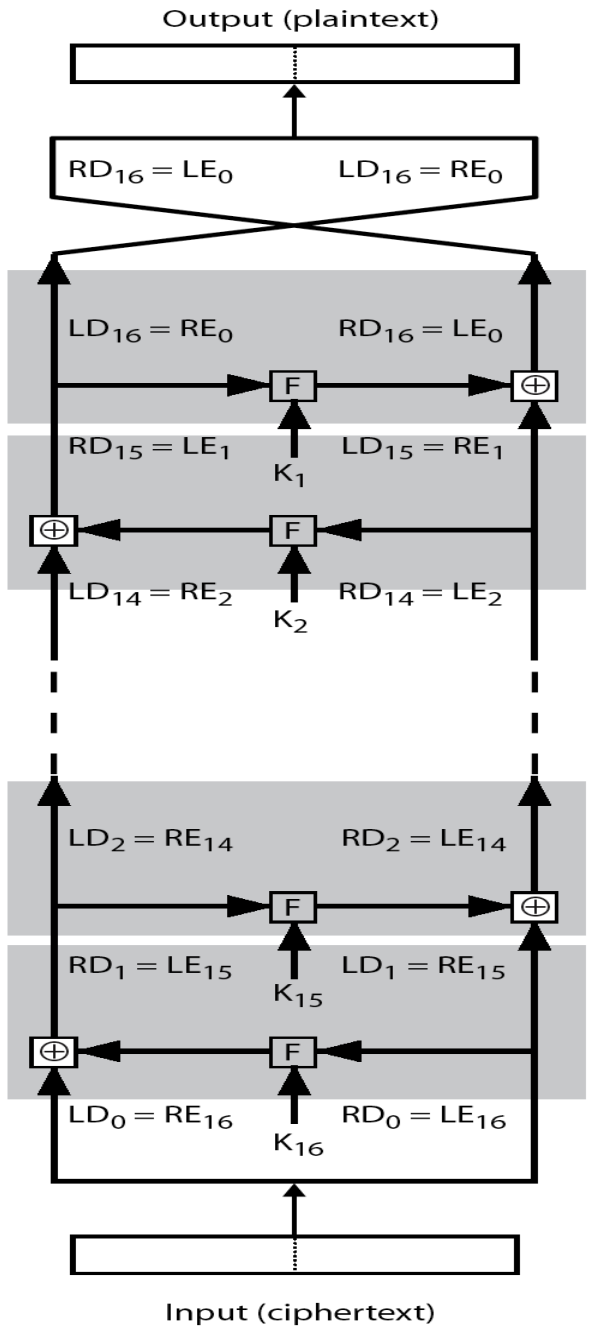


- The process of decryption with a Feistel cipher, is essentially the same as the encryption process.

- The rule is as follows: Use the ciphertext as input to the algorithm, but use the subkeys *Ki* in reverse order. That is, use *Kn* in the first round, K_{n-1} in the second round, and so on until *K*1 is used in the last round.

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





DATA ENCRYPTION STANDARD

History of DES

In 1971, IBM developed an algorithm, named LUCIFER which operates on a block of 64 bits, using a 128-bit key

Walter Tuchman, an IBM researcher, refined LUCIFER and reduced the key size to 56-bit, to fit on a chip.

In 1973, the National Bureau of Standards (NBS) issued a request for proposals for a national cipher standard. IBM submitted the modified LUCIFER

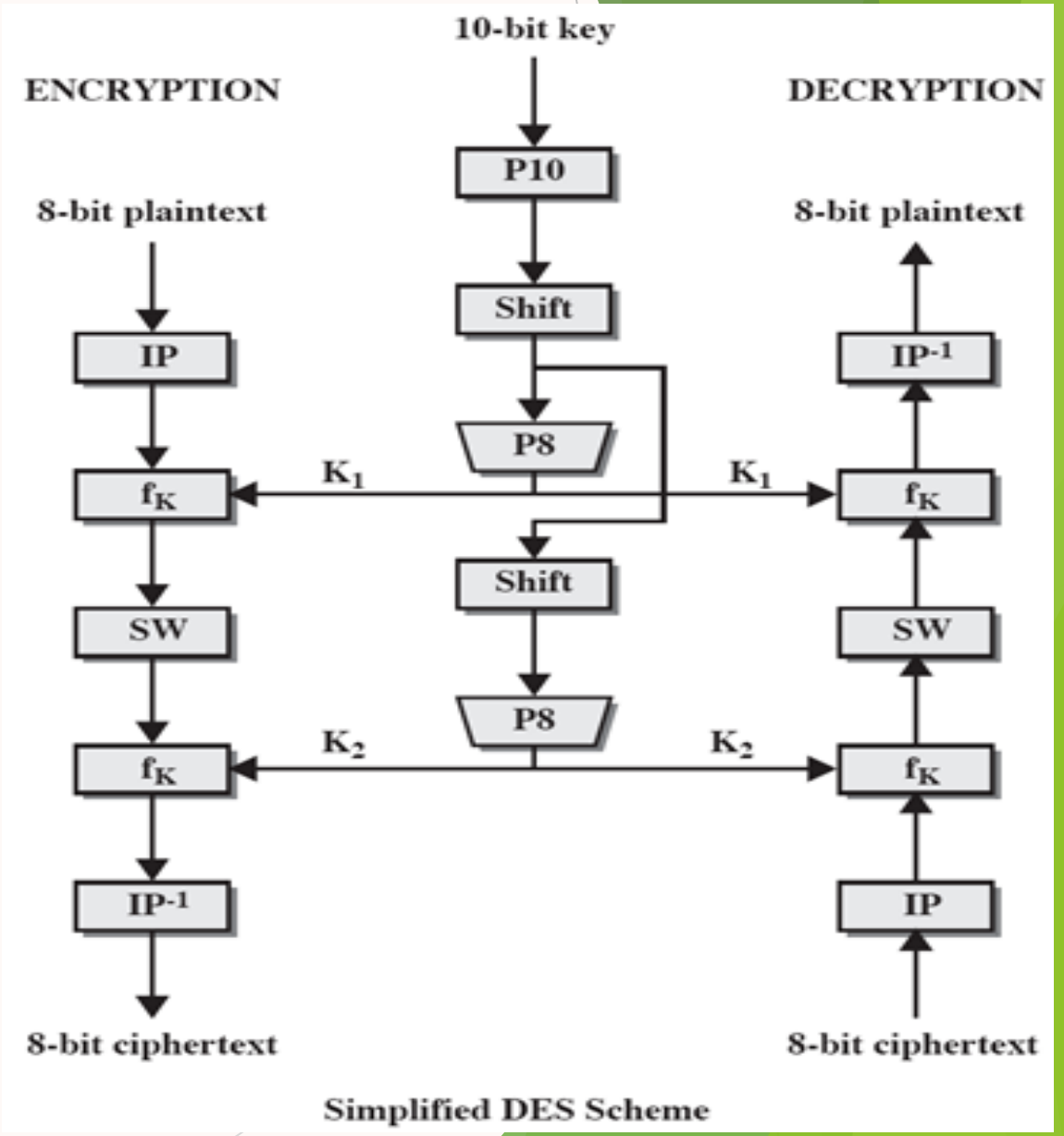
In 1977, the results of Tuchman's project of IBM was adopted as the Data Encryption Standard by NSA (NIST).

Simplified DES (SDES)

- Simplified DES, developed by Professor Edward Schaefer of Santa Clara University [SCHA96], is an educational rather than a secure encryption algorithm. It has similar properties and structure to DES with much smaller parameters.

- Like DES, this algorithm is also a block cipher. It operates on 8-bit message blocks with a 10-bit key size.

- The encryption algorithm involves five functions: and initial permutation (IP), a complex function labeled f_k, which involves both permutations & substitution operations depending on the key input, a single permutation function (SW) that switches the two halves of the data, the function f_k again and finally a permutation function that is inverse of the IP i.e. IP^{-1}.



Simplified DES Scheme

- The key is choosen to be 10-bit length from which two 8-bit subkeys are generated. The initial 10-bit key is subjected to a permutation (P10) followed by a shift operation. The output of this shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first key (k_1) and also feeds into another shift and another instance of P8 to produce the second subkey (k_2).
- The encryption algorithm can be written as:

$$\textbf{Ciphertext} = IP^{-1}(f_{k2}(SW(f_{k1}(IP(\text{plaintext})))))$$

Where $K_1 = P8(shift(p10(key)))$

$K_2 = P8(shift(shift(p10(key))))$

Decryption is also shown and can be given as:

$$\textbf{Plaintext} = IP^{-1}(f_{k1}(SW(f_{k2}(IP(\text{ciphertext})))))$$

SDES Key Generation

First, a 10-bit key shared between sender and receiver is used and initially passed through a permutation P10, Where P10 is a permutation with table:

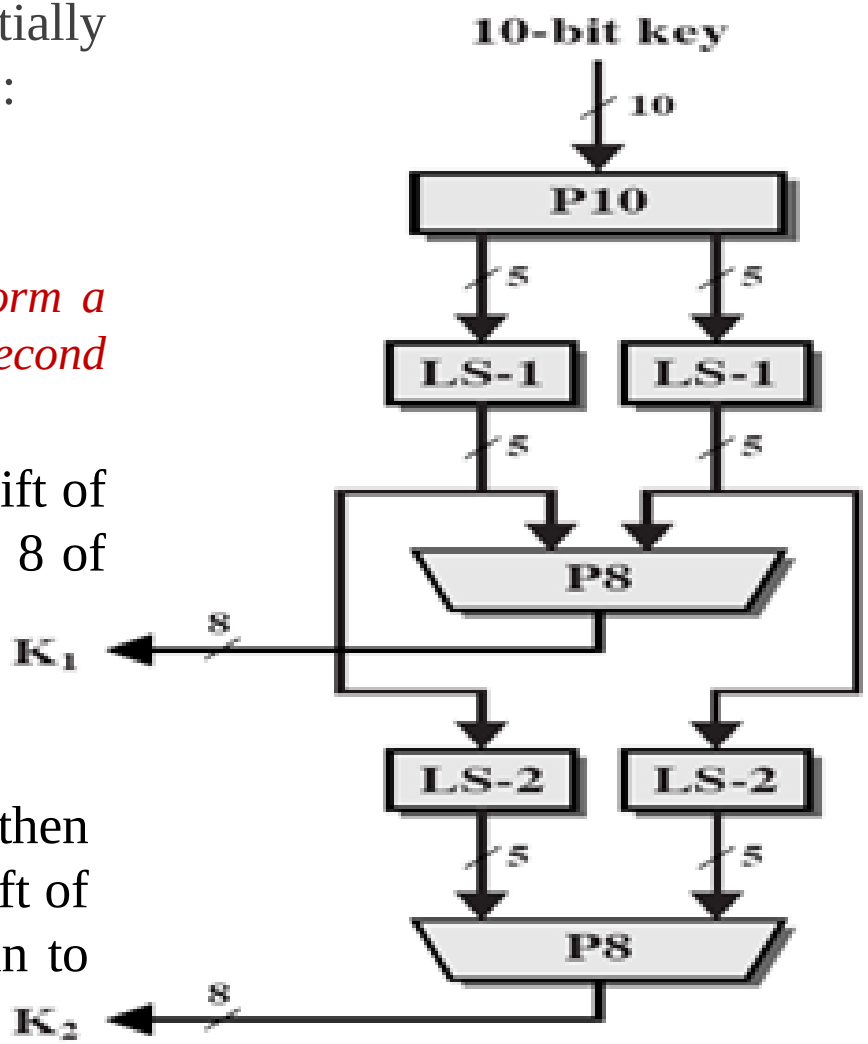
P10									
3	5	2	7	4	10	1	9	8	6

For example, the key (1010000010) is permuted to (1000001100). Next, perform a circular left shift (LS-1), or rotation, separately on the first five bits and the second five bits. In our example, the result is (00001 11000).

LS-1 is a circular left shift of 1 bit position, and LS-2 is a circular left shift of 2 bit positions. P8 is another permutation which picks out and permutes 8 of the 10 bits according to the following rule:

P8							
6	3	7	4	8	5	10	9

The result is subkey 1 (K_1) *In our example, this yields (10100100)* and then the outputs from the two LS-1 functions are taken and a circular left shift of 2 bit positions is performed on each string and then P8 is applied again to produce K_2.



We then go back to the pair of 5-bit strings produced by the two LS-1 functions and perform a circular left shift of 2 bit positions on each string. In our example, the value (00001 11000) becomes (00100 00011). Finally, P8 is applied again to produce K2. In our example, the result is (01000011).

SDES Encryption

The input to algorithm is an 8-bit block of plaintext which is permuted using the IP function. The inverse to this function IP^{-1} is applied towards the end of algorithm. IP is the initial permutation and IP^{-1} is its inverse.

IP							
2	6	3	1	4	8	5	7

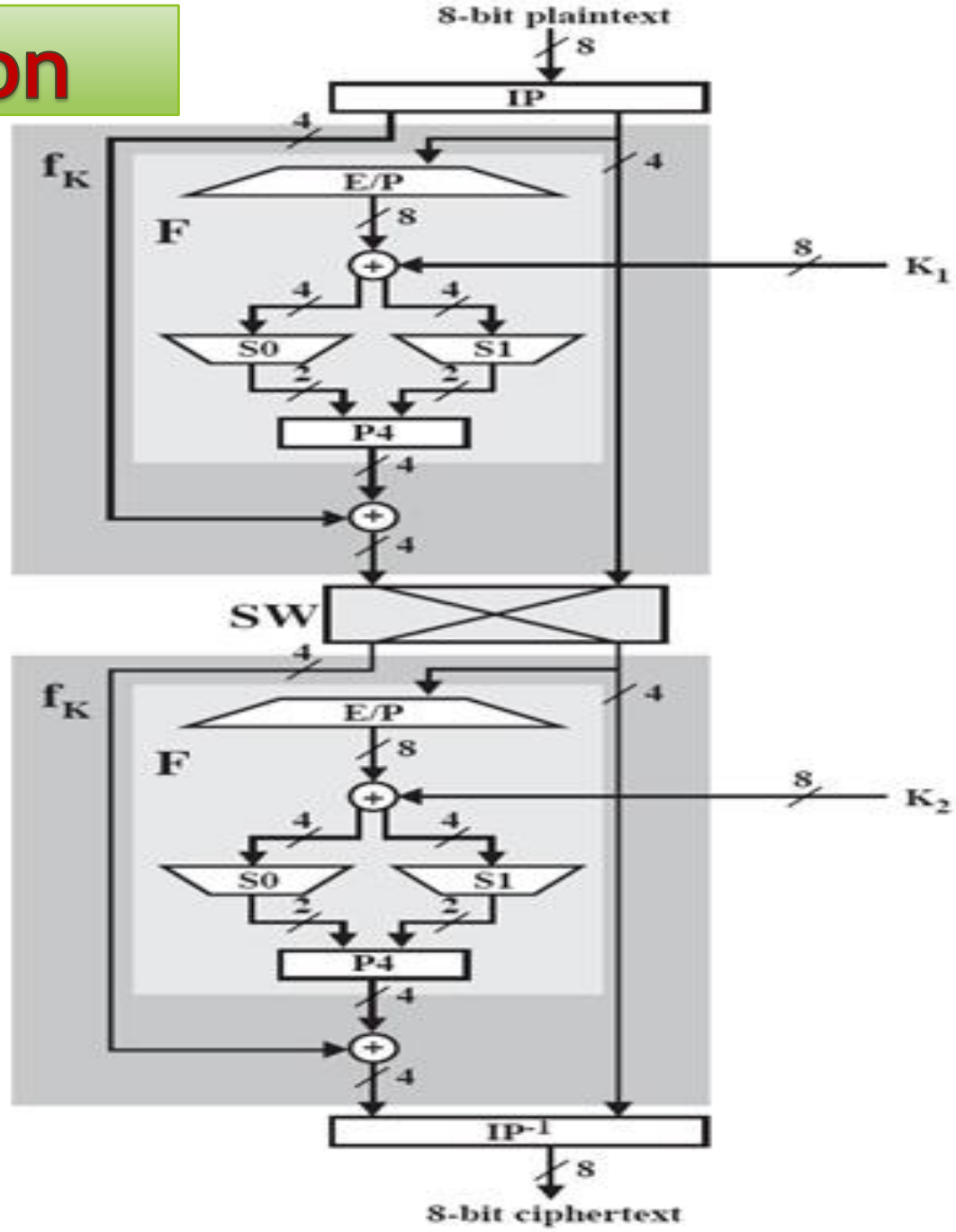
IP^{-1}							
4	1	3	5	7	2	8	6

The function f_k is the most complex component of S-DES. It consists of a combination of permutation and substitution functions.

$$f_K(L, R) = (L \oplus F(R, SK), R)$$

Where L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to f_K, and let F be a mapping from 4-bit strings to 4-bit strings. SK is a subkey and $\oplus$ is the bit-by-bit exclusive-OR function.

For example, suppose the output of the IP stage is (10111101) and F(1101, SK) = (1110) for some key SK.
Then $f_K(10111101) = (01011101)$
because $(1011) \oplus (1110) = (0101)$.



Simplified DES Encryption Detail

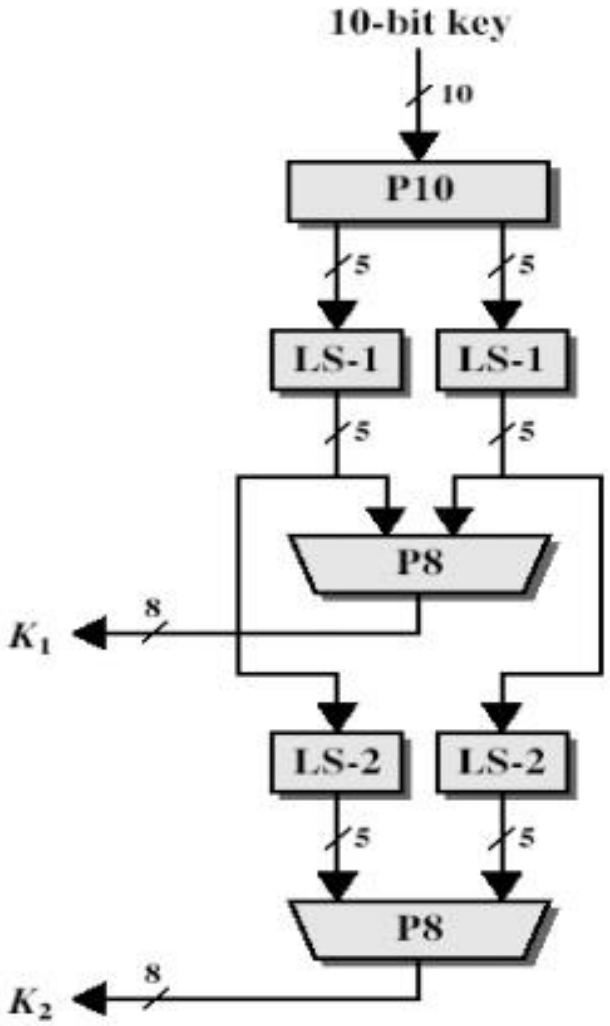
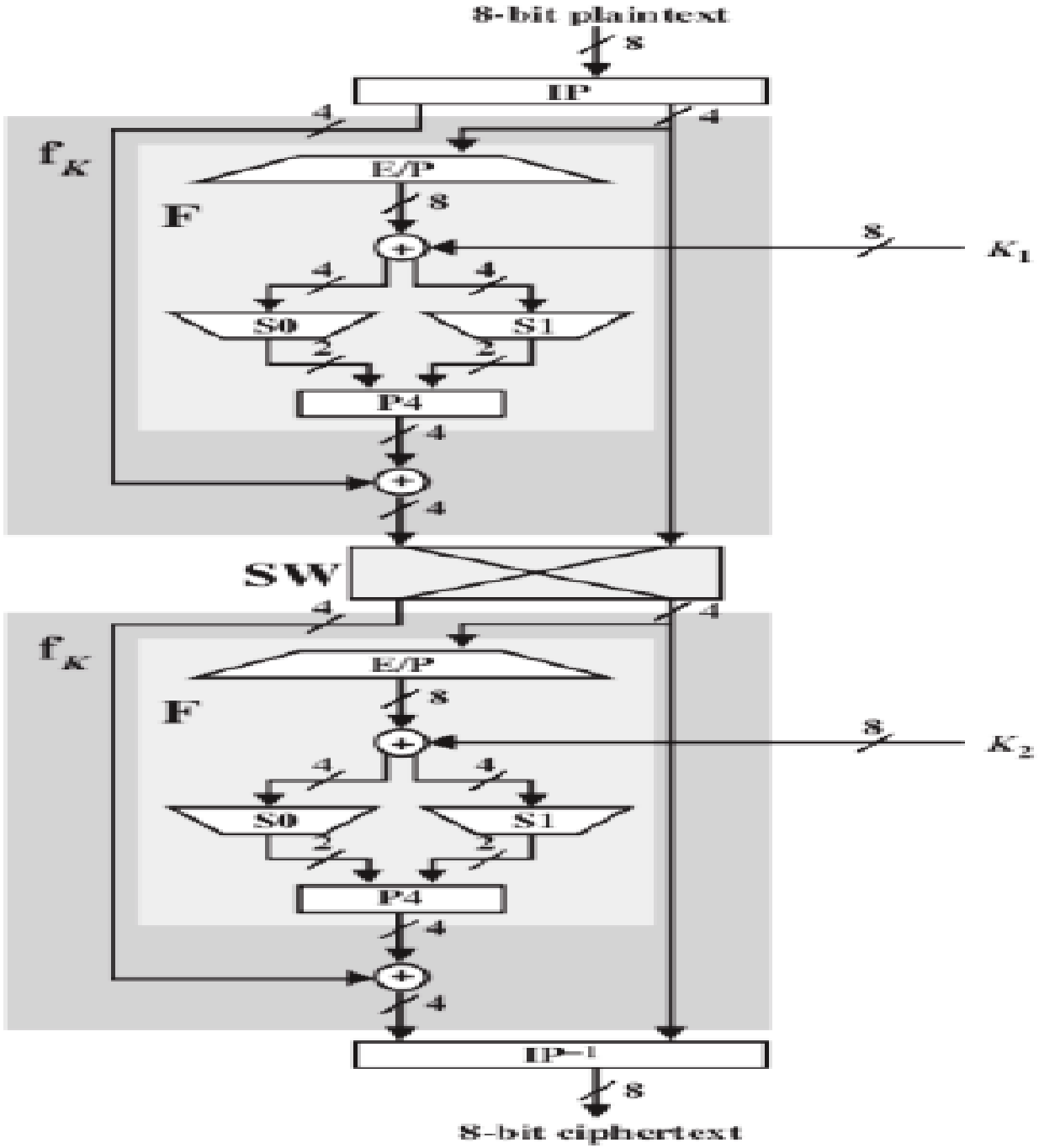


Figure: key generation for S-DES



- For the mapping function F, input is a 4-bit number (*n*1*n*2*n*3*n*4). The first operation is an expansion/permutation operation.

E/P							
4	1	2	3	2	3	4	1

- The 8-bit subkey K1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18}) is added to this value using XOR.
- The first 4 bits are fed into the S-box S0 to produce a 2-bit output, and the remaining 4 bits are fed into S1 to produce another 2-bit output. These two boxes are defined as follows:

S0

1	0	3	2
3	2	1	0
0	2	1	3
3	1	3	2

S1

0	1	2	3
2	0	1	3
3	0	1	0
2	1	0	3

The S-boxes operate as follows. The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the Sbox. The entry in that row and column, in base 2, is the 2-bit output. Next, the 4 bits produced by S0 and S1 undergo a further permutation P4. The output of P4 would be the output of function F.

P4			
2	4	3	1

The Switch function interchanges the left and right 4 bits so that the second instance of f_K operates on a different 4 bits. For second instance all other parameters remain same, but the key is K_2

Example for S-DES

Assume 10-bit key, K is: 1010000010.

Then steps for generating the two 8-bit round keys, K1 & K2, are:

1. Rearrange K using P10: 1000001100
2. Left shift by 1 position both the left and right halves: 00001 11000
3. Rearrange the halves with P8 to produce K1: 10100100
4. Left shift by 2 positions the left and right halves: 00100 00011
5. Rearrange the halves with P8 to produce K2: 01000011

K1 and K2 are used as inputs in the encryption and decryption stages.

Assume a 8-bit plaintext, P: 01110010, then the steps for encryption are:

1. Apply the initial permutation, IP, on P: 10101001
2. Assume the input from step 1 is in two halves, L and R: L=1010, R=1001
3. Expand and permutate R using E/P: 11000011
4. XOR input from step 3 with K1: 10100100 XOR 11000011 = 01100111
5. Input left halve of step 4 into S-Box S0 and right halve into S-Box S1:
   a. For S0: 0110 as input: b1,b4 for row, b2,b3 for column
   b. Row 00, column 11 -> output is 10
   c. For S1: 0111 as input:
   d. Row 01, column 11 -> output is 11

6. Rearrange outputs from step 5 (1011) using P4: 0111
7. XOR output from step 6 with L from step 2: 0111 XOR 1010 = 1101
8. Now we have the output of step 7 as the left half and the original R as the right half. Switch the halves and move to round 2: 1001 1101
9. E/P with right half: E/P(1101) = 11101011
10. XOR output of step 9 with K2: 11101011 XOR 01000011 = 10101000
11. Input to s-boxes:

    a. For S0, 1010

    b. Row 10, column 01 -> output is 10

    c. For S1, 1000

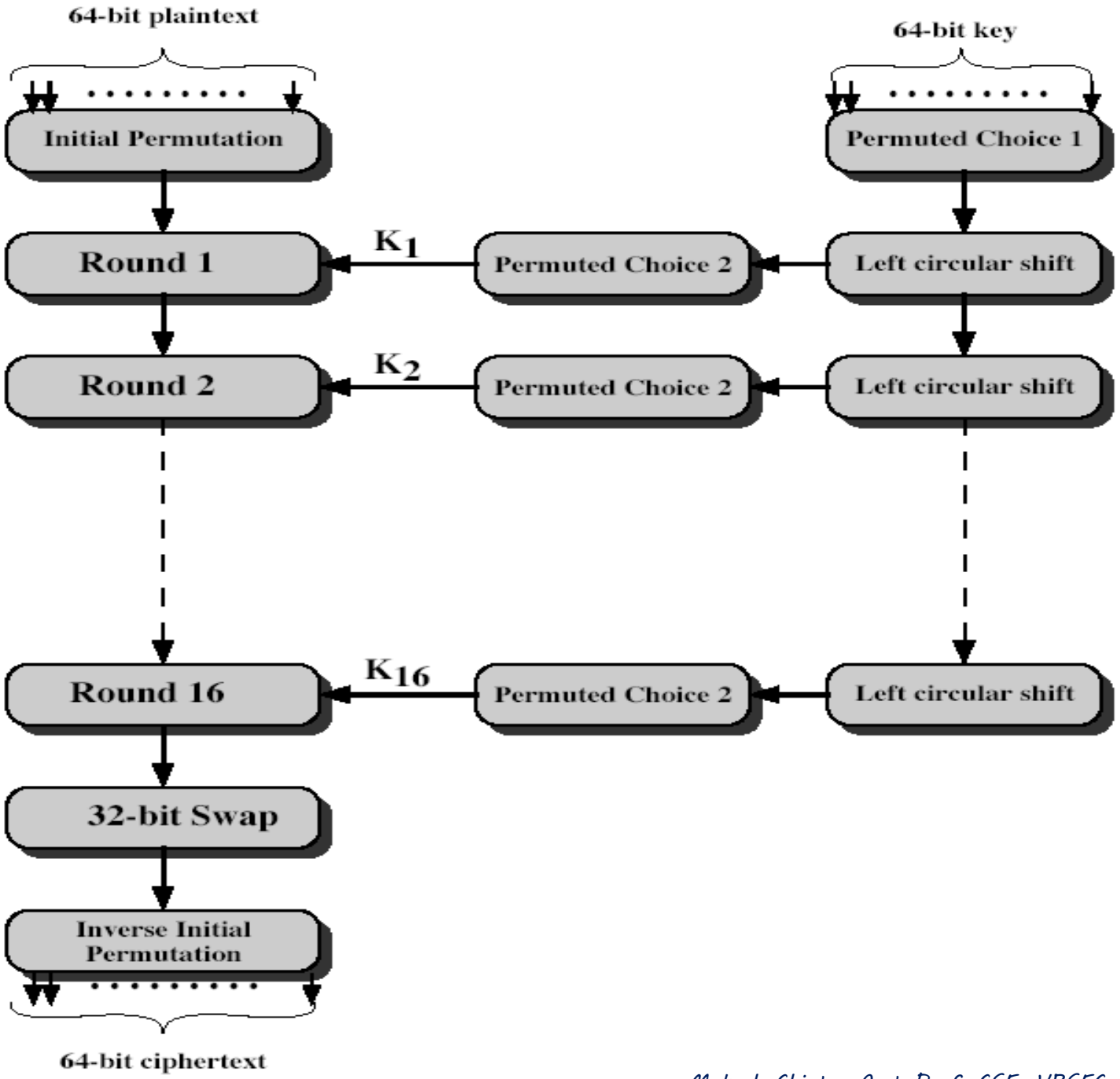
    d. Row 10, column 00 -> output is 11
12. Rearrange output from step 11 (1011) using P4: 0111
13. XOR output of step 12 with left halve from step 8: 0111 XOR 1001 = 1110
14. Input output from step 13 and right halve from step 8 into inverse IP
15. a. Input is 1110 1101

So our encrypted result of plaintext 01110010 with key 1010000010 is: 01110111

Data Encryption Standard

- The Data Encryption Standard is a symmetric-key algorithm for the encryption of electronic data. Although now considered insecure, it was highly influential in the advancement of modern cryptography.
- DES uses the two basic techniques of cryptography - confusion and diffusion. At the simplest level, diffusion is achieved through numerous permutations and confusion is achieved through the XOR operation and the S-Boxes. This is also called an S-P network.
- DES was approved as a federal standard in November 1976, and published on 15 January 1977 as FIPS PUB 46, authorized for use on all unclassified data.
- On 26 May 2002, DES was finally superseded by the Advanced Encryption Standard (AES). On 19 May 2005, FIPS 46-3 was officially withdrawn, but NIST has approved Triple DES through the year 2030 for sensitive government information.
- DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm.

http://kathrynneugent.com/animation.html



The main phases of DES structure are:

- **Initial Permutation (IP):** The plaintext block undergoes an initial permutation. 64 bits of the block are permuted.
- **A Complex Transformation:** 64 bit permuted block undergoes 16 rounds of complex transformation. Subkeys are used in each of the 16 iterations.
- **32-bit swap:** The output of 16th round consists of 64bits that are a function of input plain text and key.32 bit left and right halves of this output is swapped.
- **Inverse Initial Permutation (IP^{-1}):** The 64 bit output undergoes a permutation that is inverse of the initial permutation.

(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

- The permutation **X = IP(*M*)**
- The inverse permutation

  $Y = IP^{-1}(X) = IP^{-1}(IP(M))$

  The original ordering is restored

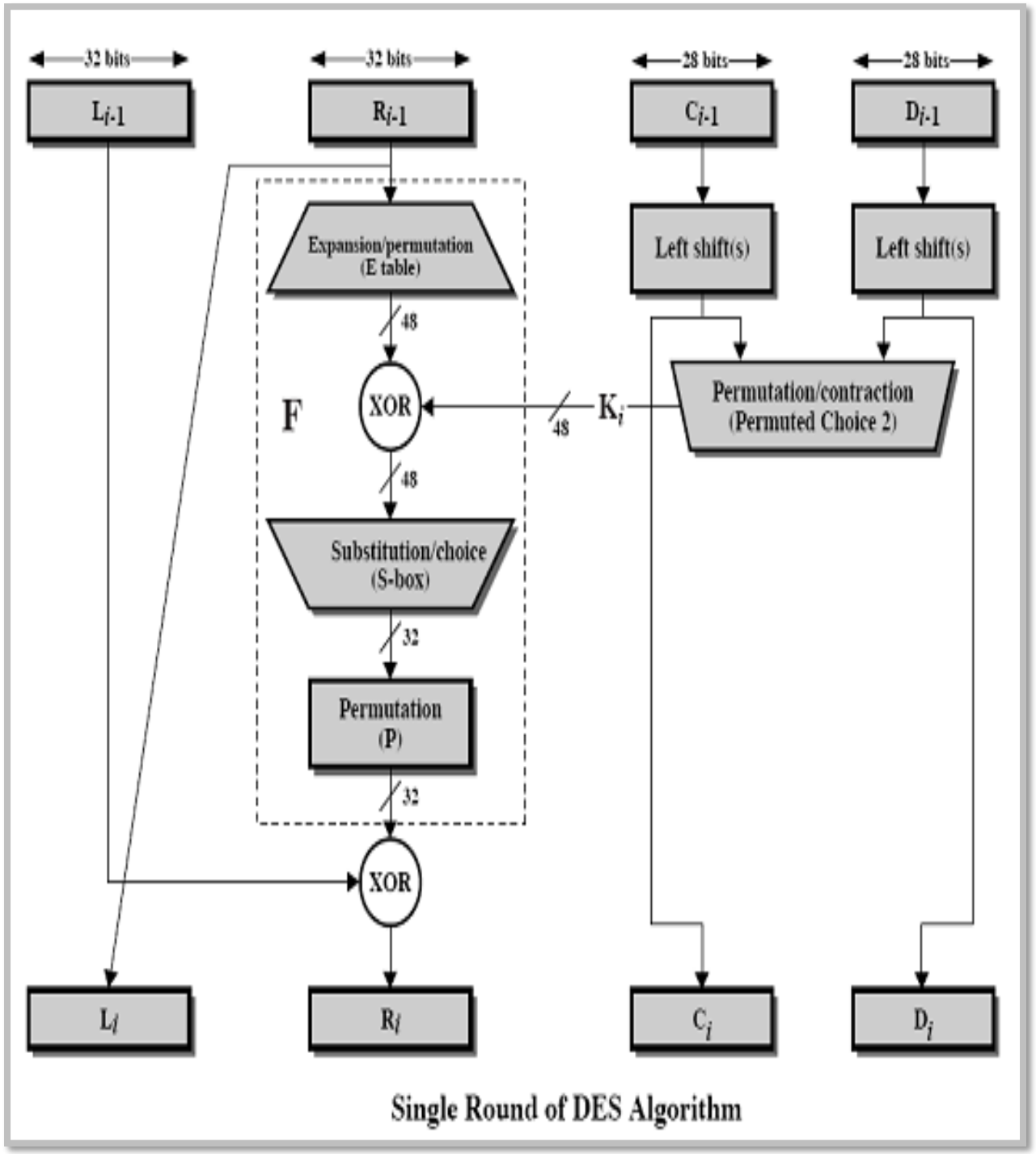
(c) Expansion Permutation (E)

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

- The E-Box: 32 bits → 48 bits
- 16 bits are reused.
- The P-Box: Output of the S-Boxes is permuted



Single Round of DES Algorithm

- The 64bit permuted input passes through 16 iterations, producing an intermediate 64-bit value at the conclusion of each iteration.
- The left and right halves of each 64 bit intermediate value are treated as separated 32-bit quantities labeled L (left) and R (Right). The overall processing at each iteration is given by following steps, which form one round in an S-P network.

$$Li = Ri\text{-}1.$$

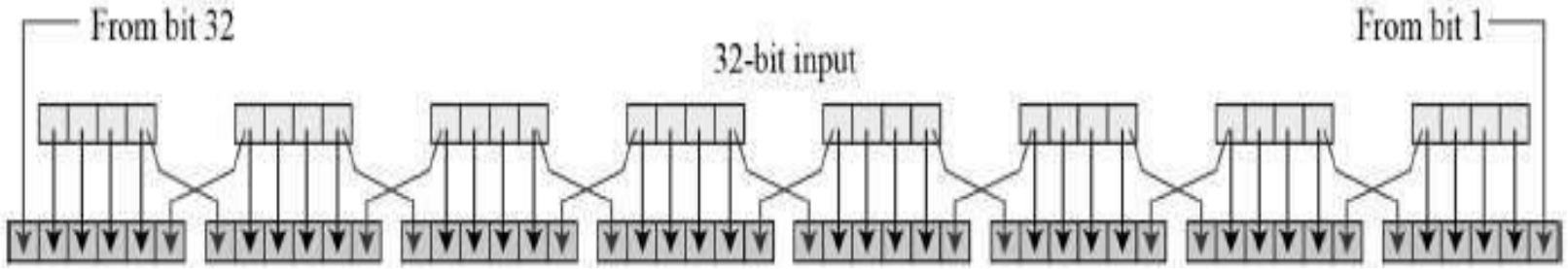
$$Ri = L\ i\text{-}1 \oplus F(R\ i\text{-}1, Ki)$$

Where F is described as $P(S(E(R(i\text{-}1))\oplus K(i)))$

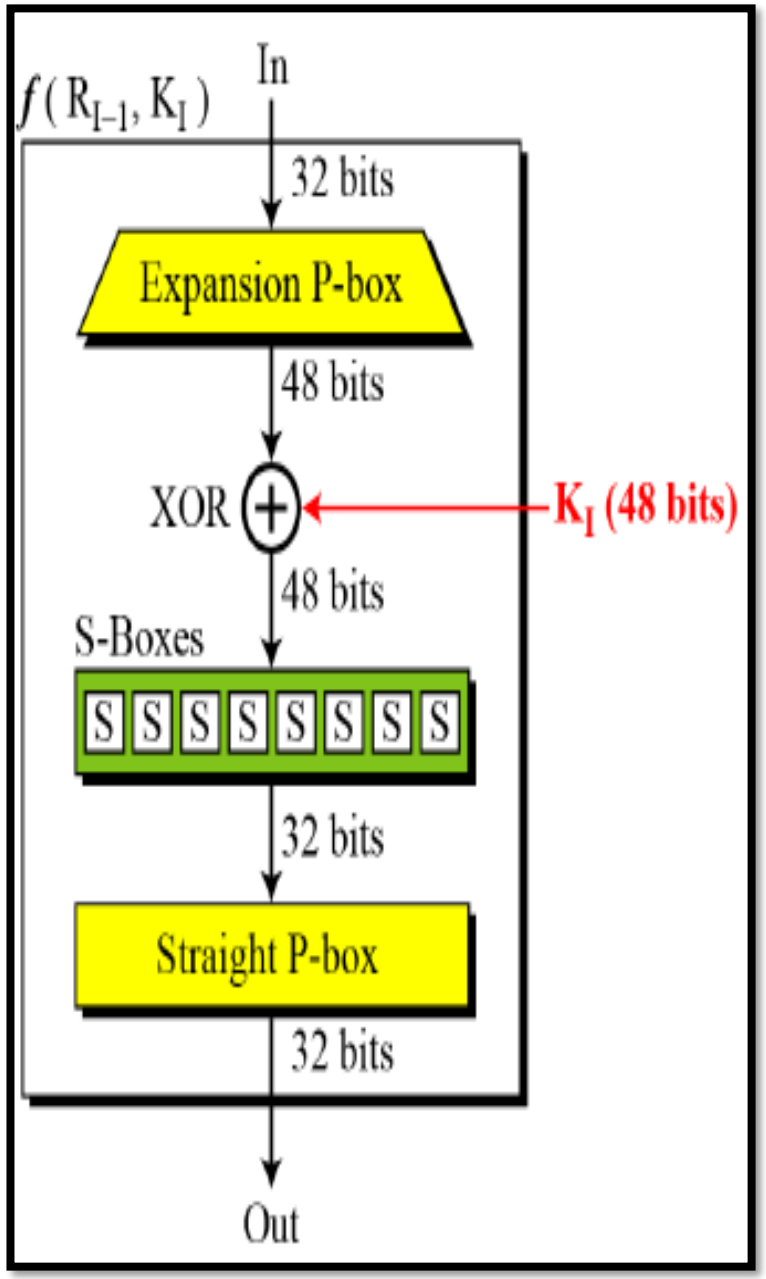
- The left hand output of an iteration (Li) is equal to the right hand input to that iteration Ri-1. The right hand output Ri is exclusive OR of Li-1 and a complex function F of Ri-1 and Ki.

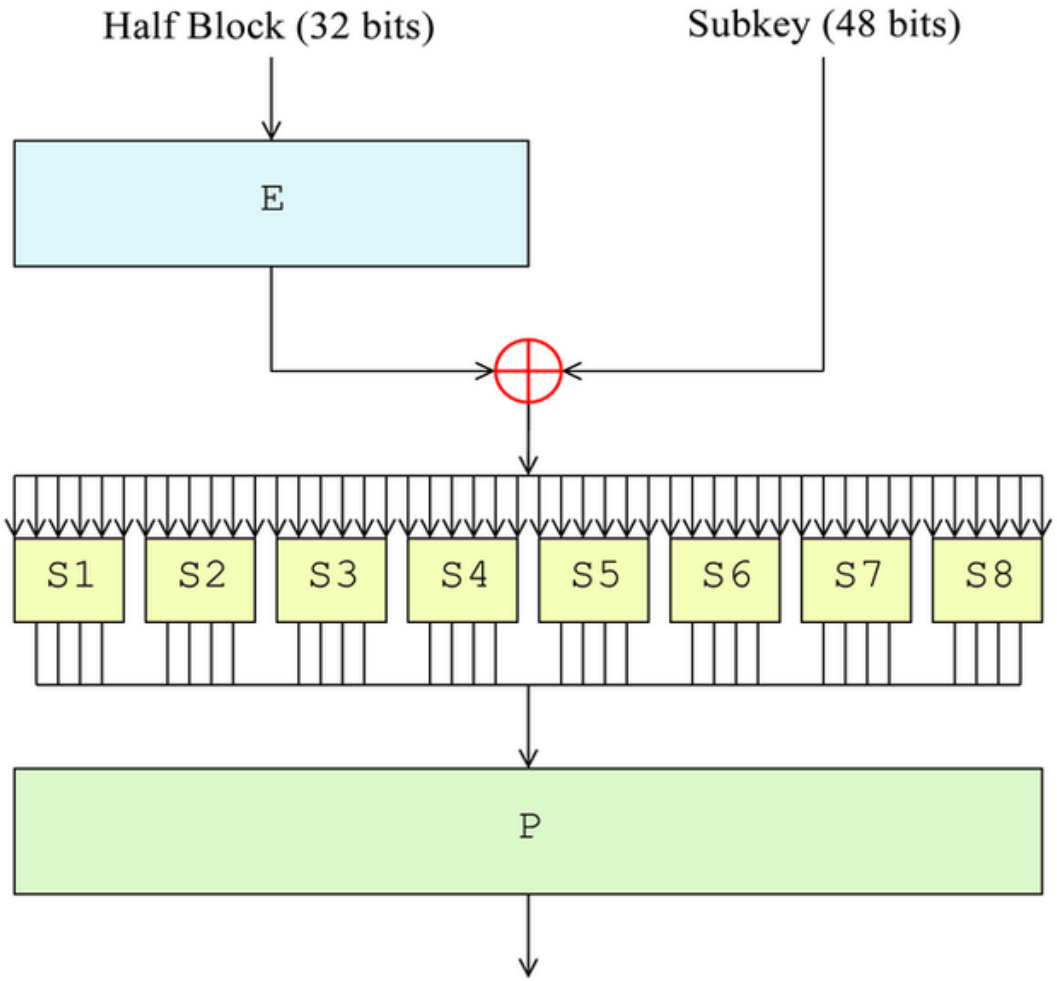
DES Round Function

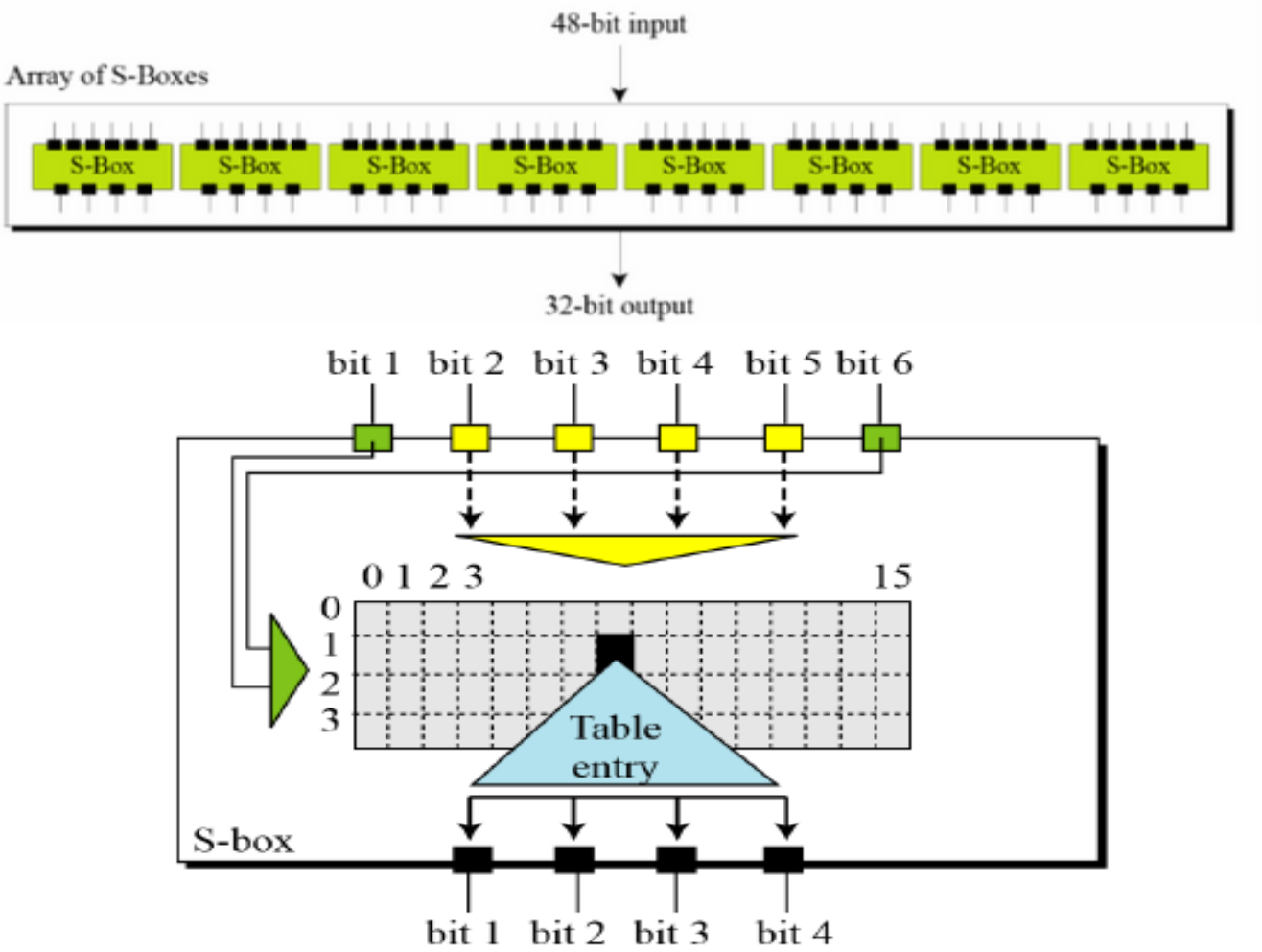
- The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.
- **Expansion Permutation Box (P-BOX)** – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits.



- **XOR (Whitener).** – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** – The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.
- **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation according to a fixed rule.







- 8 S-Boxes are used for the reduction of bits from 48bit – 32bit. for each 6-bit input, a 4-bit output is produced which results in reduction.

- First and last bit is used to select the row from the table and the 2nd, 3rd, 4th, 5th bits are used to select the column from the table resulting in a 4-bit number being selected.

S1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

S2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
1	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
2	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
3	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9

S3

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
1	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
2	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
3	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C

S4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
1	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
2	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
3	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E

S5

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
1	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
2	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
3	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3

S6

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
1	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
2	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
3	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D

S7

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
1	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
2	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
3	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C

S8

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
1	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
2	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
3	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

DES S-Boxes

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Input : 011001

- The row is **01** (row 1)
- The column is **1100** (column 12)
- Output is **1001**

DES Sub-Key Generation

- DES operates on the 64-bit blocks using *key* sizes of 56- bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64)
- The 64-bit key is permuted according to the following table, **PC-1**. only 56 bits of the original key appear in the permuted key.

PC-1

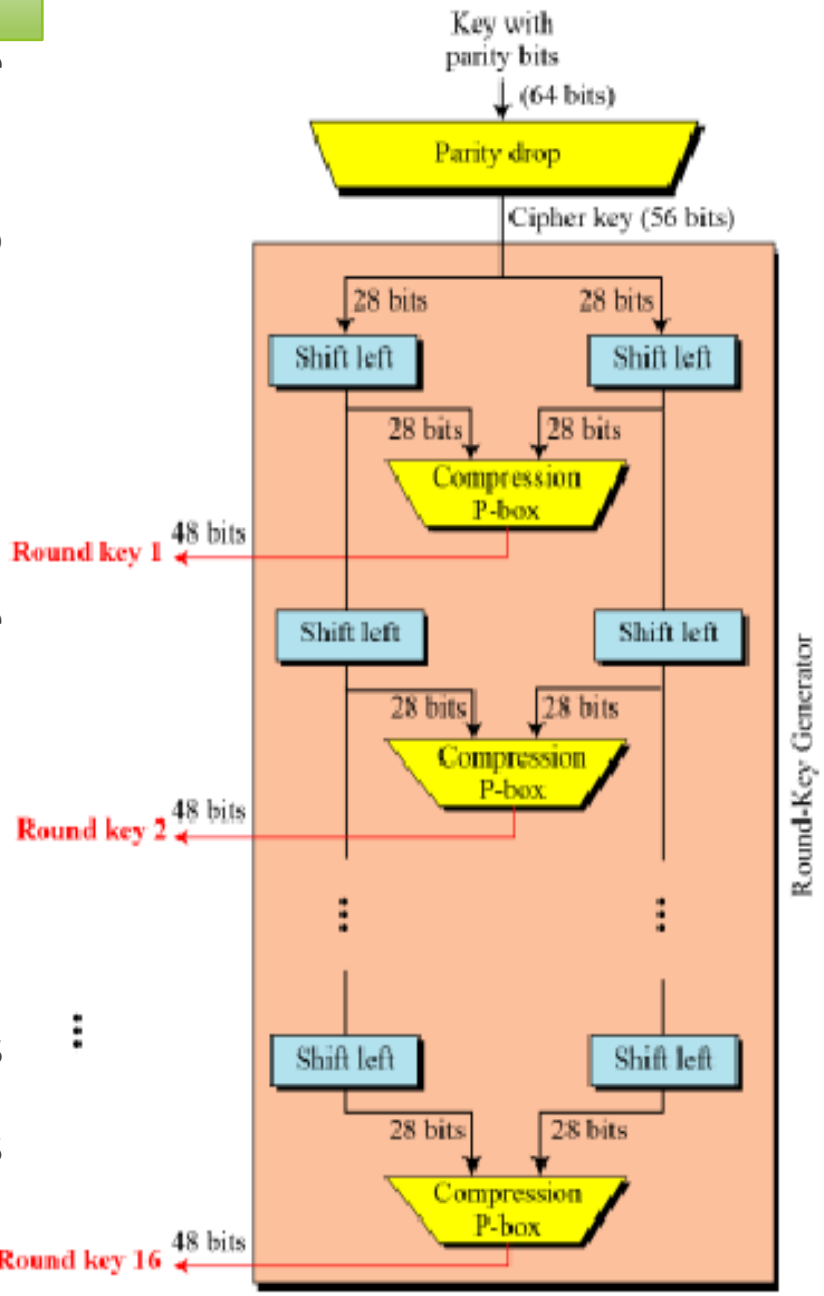
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

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

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

- The two 28-bit quantities are then subjected to successive circular left shifts of different sizes before the subkey for each round is determined from them. These circular left shifts, one of which is applied before the first subkey is taken, are in order of the following sizes:

1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1



Strength of DES

- The process of decryption with DES is essentially the same as the encryption process: no different algorithm is used.
- The ciphertext is used as input to the DES algorithm and the keys are used in the reverse order i.e. K_{16} in the first iteration, K_{15} on the second iteration and so on until k_1 is used on the sixteenth and last iteration.
- DES exhibits a strong **Avalanche effect**, which is "*A small change of plaintext or key produces a significant change in the cipher text*".
- Concern about the strength of DES falls into two categories
  - Strength of algorithm itself
  - Use of 56- bit key.
- Though many attempts were made over the years to find and exploit weaknesses in the algorithm, none of them were successful in discovering any fatal weakness in DES.

The Avalanche Effect in DES

Plaintext 1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
Plaintext 2	10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
Key	00000001	1001011	0100100	1100010	0011100	0011000	0011100	0110010

Change in 1-bit of Plaintext

(a) Change in Plaintext	
Round	Number of bits that differ
0	1
1	6
2	21
3	35
4	39
5	34
6	32
7	31
8	29
9	42
10	44
11	32
12	30
13	30
14	26
15	29
16	34

plaintext	01101000	10000101	00101111	01111010	00010011	01110110	11101011	10100100
Key 1	1110010	1111011	1101111	0011000	0011101	0000100	0110001	11011100
Key 2	0110010	1111011	1101111	0011000	0011101	0000100	0110001	11011100

Change in 1-bit of Key

(b) Change in Key	
Round	Number of bits that differ
0	0
1	2
2	14
3	28
4	32
5	30
6	32
7	35
8	34
9	40
10	38
11	31
12	33
13	28
14	26
15	34
16	35

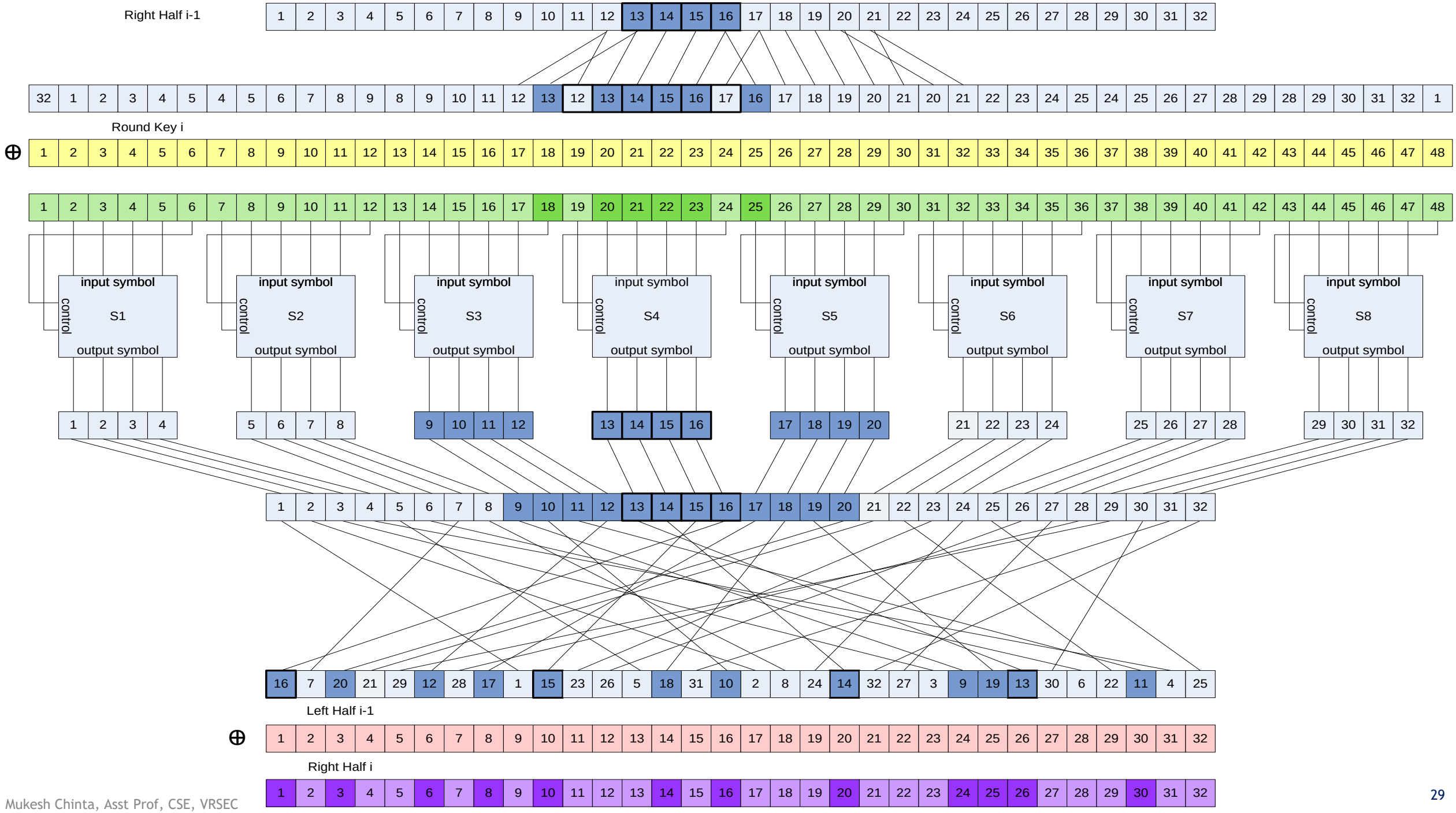
Usage of 56-bit Keys

- If the key length is 56-bit, we have **$2^{56} = 7.2 \times 10^{16}$ keys**

- Diffie and Hellman then outlined a "brute force" attack on DES. They proposed a special purpose "parallel computer using one million chips to try one million keys each" per second, and estimated the cost of such a machine at $20 million.

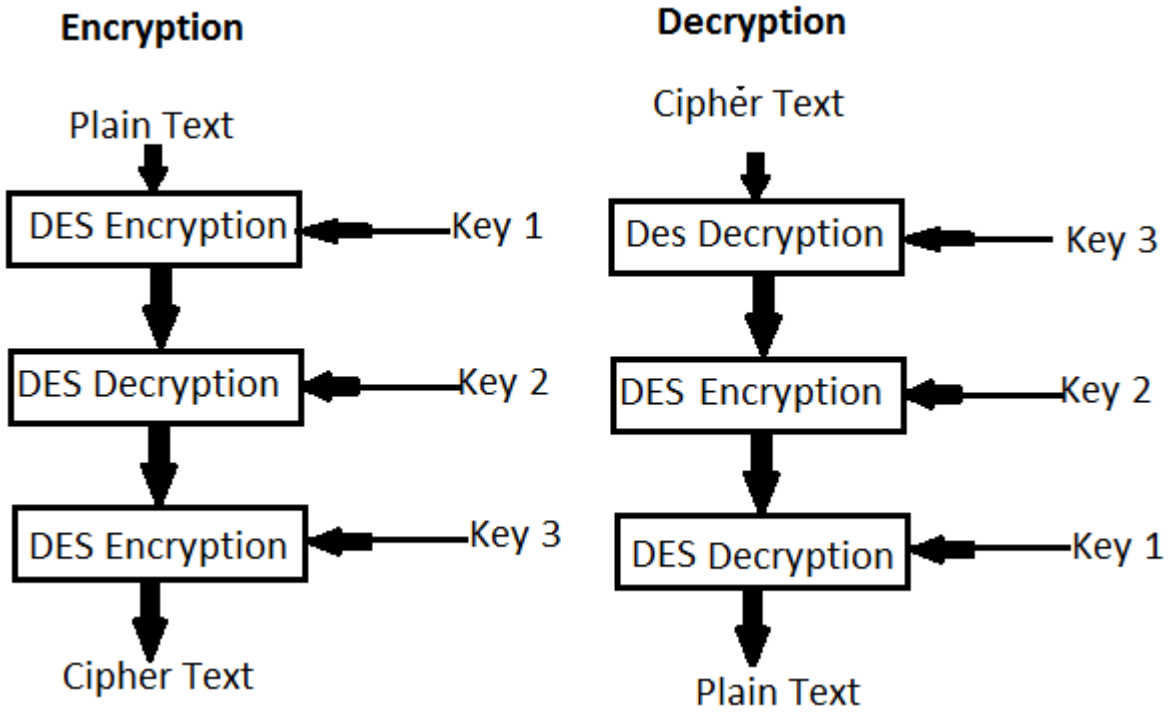
- In 1998, under the direction of John Gilmore of the EFF (**Electronic Frontier Foundation)** team spent $220,000 and built a machine that can go through the entire 56-bit DES key space in an average of 4.5 days. On July 17, 1998, they announced they had cracked a 56-bit key in 56 hours. The computer, called Deep Crack, uses 27 boards each containing 64 chips, and is capable of testing 90 billion keys a second.

- The two analytical attacks on DES are Differential cryptanalysis and Linear cryptanalysis. Both make use of Known plaintext-ciphertext pairs and try to attack the round structure and the S-Boxes. Recent advancements showed that using Differential cryptanalysis, DES can be broken using 2^{47} plaintext-ciphertext pairs and for linear cryptanalysis, the number is even reduced to 2^{41}.


Right Half i-1
Round Key i
input symbol
control
S1
S2
S3
S4
S5
S6
S7
S8
output symbol
Left Half i-1
Right Half i

Triple DES (3DES)

- Triple DES applies the Data Encryption Standard (DES) cipher algorithm three times to each data block.
- When it was discovered that a 56-bit key of DES is not enough to protect from attacks, TDES was chosen as a simple way to enlarge the key space without a need to switch to a new algorithm.
- Triple DES simply extends the key size of DES by applying the algorithm three times in succession with three different keys. It uses a key length of 168 bits: three 56-bit DES keys



The encryption algorithm is:

$$ciphertext = E_{K3}(D_{K2}(E_{K1}(plaintext)))$$

i.e., encrypt with K1, DES decrypt with K2, then encrypt with K3.

Decryption is the reverse:

$$plaintext = D_{K1}(E_{K2}(D_{K3}(ciphertext)))$$

i.e., decrypt with K3, encrypt with K2, then decrypt with K1.

Each triple encryption encrypts one block of 64 bits of data. In each case the middle operation is the reverse of the first and last. This improves the strength of the algorithm when using keying option 2, and provides backward compatibility with DES with keying option 3.

Other Popular Symmetric Algorithms

Blowfish

- Blowfish by **Bruce Schneier**, author of Applied Cryptography, is considered as a highly rated encryption algorithm in terms of security, with different structure and Functionality than the other mentioned encryption algorithms. Blowfish is a fast, compact, and simple block encryption algorithm with variable length key allowing a tradeoff between speed and security. Blowfish is a **public domain algorithm** (unpatented) and is used in the SSL and other program.
- Blowfish is a block cipher that uses a **64 bit plain text** with **16 rounds**, allowing a variable key length, up to **448 bits**, permuted into 18 sub-keys each of 32-bit length and can be implemented on 32- or 64-bit processors. It also contains **4 S-boxes** and **same algorithm** is used in reverse for decryption.

IDEA

- James L. Massey and Xuejia Lai (Zurich, Switzerland) in 1990 developed an encryption algorithm named as **International Data Encryption Algorithm (IDEA).** It is fairly fast, considered secure, and is also resistant to both linear and differential analysis. IDEA is considered one of the secure block ciphers offered in public domain in last decades.
- *IDEA* is symmetric key algorithm based on the concept of Substitution-Permutation Structure. It is a block cipher that uses a **64 bit plain text** with **8 rounds** and a Key Length of **128-bit** permuted into 52 sub-keys each of 128-bits. It **does not contain S-boxes** and same algorithm is used in reversed for decryption.

Comparison of Symmetric Algorithms

Summary of Symmetric Algorithms Architecture

	Algorithm Structure	Plain Text/Cipher Text Length	Key Size	# S boxes	# of Rounds
DES	Festial structure	64 bits	56	8	16
3DES	Festial structure	64 bits	168	8	48
Blowfish	Festial structure	64 bits	128-448	4	16
IDEA	Substitution-Permutation Structure	64 bits	128	N/A	8
TEA	Festial structure	64 bits	128	N/A	64 (32 cycles)
CAST	Festial structure	64 bits	40-128	4	12 – 16
Rijndael	Festial structure	128 Bits	128,192,256	1	10,12,14
RC6	Festial structure	128 Bits	128,192,256	N/A	20
Serpent	Festial structure	128 Bits	128,192,,256	8	32
Twofish	Festial	128 Bits	128,192,256	4	16
MARS	Festial	128 Bits	128-448	1	32