Data Encryption Standard (DES)

- DES is block cipher, symmetric algorithm: The same algorithm and key are used for both encryption and decryption (except for minor differences in the key schedule).

- Public key algorithms such as RSA (Rivest, Shamir, Adleman), which use public keys that can be public knowledge. However RSA security relies on its superior mathematical complexity.

- In contrast DES uses a relatively simple algorithm but the key must be a very closely guarded secret. This implies that very elaborate key management systems must be used for DES.

- DES is now considered to be insecure for many applications. This is mainly due to the 56-bit key size being too small. In January 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes.

- There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks.

- In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES).

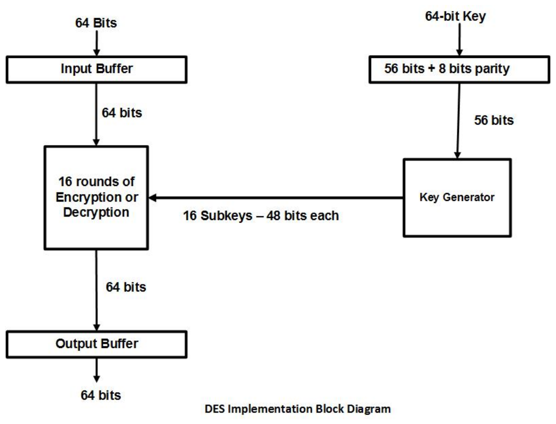
- DES is an implementation of a Feistel Cipher. It uses a 16-round Feistel structure. A 64-bit block of plaintext goes in one end of the algorithm and a 64-bit block of ciphertext comes out the other end.

- Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm function as check bits only.

- The key is usually expressed as a 64-bit number, but every 8th bit is used for parity checking and is ignored.

- Each parity-check bit is the XOR of the previous 7 bits. These parity bits are the least-significant bits of the key bytes.

- The following diagram depicts a high-level functionality of the overall DES algorithm:



DES Implementation Block Diagram

- The algorithm operates on 64-bit blocks or plaintext or ciphertext (input block can contain either plaintext or ciphertext) using a 64-bit key.

- Since each key is effectively 56 bits, there are $\mathbf{2^{56}}$ (7.2 x $10^{16)}$ possibilities for a key. Keys are selected randomly to minimize repetition.

- The encryption and decryption algorithms operate on 64-bit input block to generate 64-bit output blocks.

- The DES algorithm can be operated in three modes:

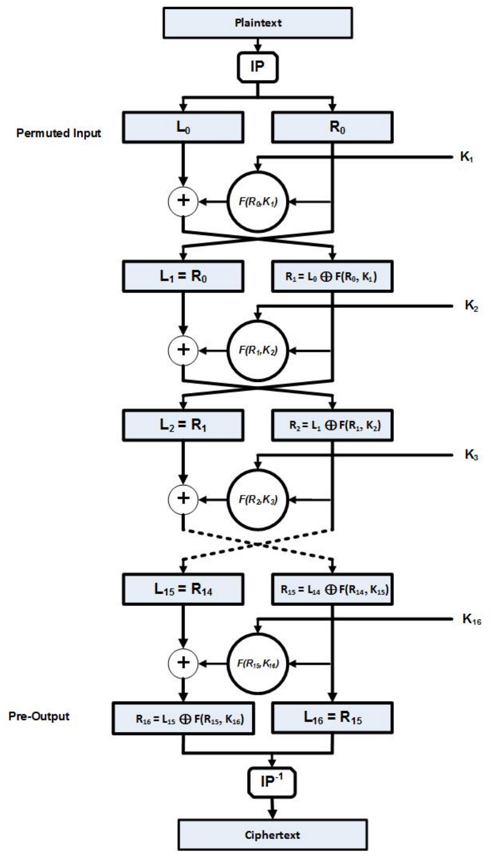
  - Electronic Code Book (ECB) mode
  - Cipher Block Chaining (CBC) mode
  - Cipher Feedback (CFB) mode

- In CBC mode, a 64-bit block of plaintext is fed into the algorithm to generate the ciphertext.

- In CBC and CFB modes, Initialization Vectors (IV) are added modulo 2 (XOR) to the plaintext and then fed into the algorithm to generate the ciphertext.

DES Enciphering

- The algorithm consists of 16 identical rounds; each round (except for the first) uses the 64-bit result produced by the previous round.
- Each round uses one of the 16-bit subkeys.
- The encryption and decryption processes are identical with one important difference: **the sequence of subkeys used for decryption is in the opposite order of that used in the encryption sequence.**
- The following diagram illustrates the 16-round DES architecture:



- The general process is mathematically defined for encryption as (very similar analysis to Feistel):

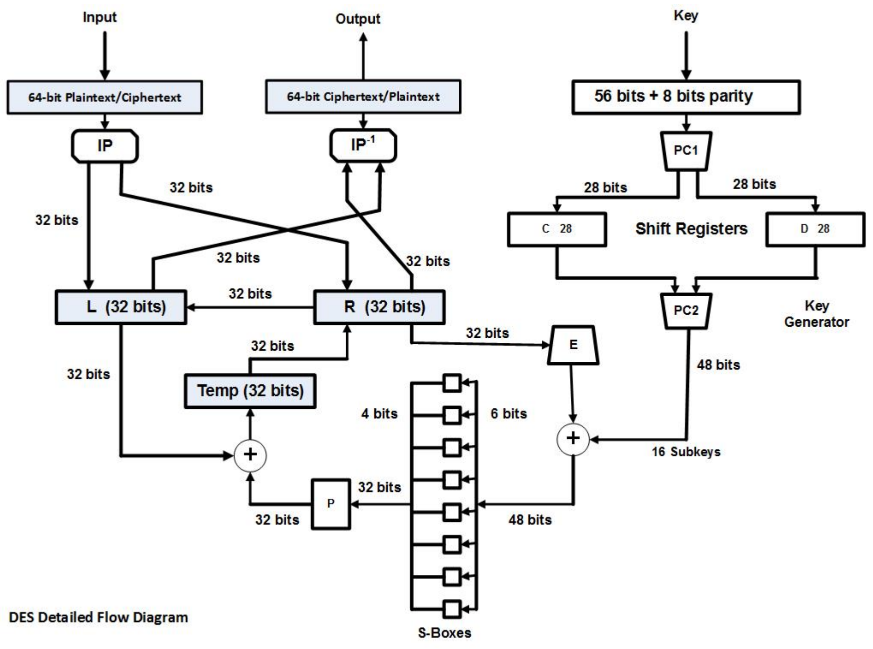
$$\mathbf{IP(x) = L_0R_0}$$
$$\mathbf{L_i = R_{i-1}}$$
$$\mathbf{R_i = L_{i-1} \oplus F(R_{i-1}, K_i)}$$
$$\mathbf{y = IP^{-1}(R_{16}\,L_{16})}$$

- Where,

IP = Initial Permutation; IP^{-1} = Inverse permutation; x = plaintext; y = ciphertext

- Note that after the last iteration, the left and right halves are not exchanged; instead the concatenated block consisting of $\mathbf{R_{16}L_{16}}$ is input to the final permutation $\mathbf{P^{-1}}$. This is necessary in order that the algorithm can be used for both, encryption and decryption.

- The following diagram provides a much more detailed description of one complete cycle of the DES algorithm:



DES Detailed Flow Diagram

- The right-hand side of the diagram shows the key generator and the left-hand side shows the 16 steps of encryption.

- There are two types of ciphers used here: **Permutation** and **Substitution** (this is referred to as a **Substitution-Permutation Network** or **SPN**).

- The **S-Box** provides the **confusion** function and the **permutation** rules (**P-Box**) implement the **diffusion** operation in SPN ciphers.

- These units are depicted in the diagram above as:
  - **IP – Initial Permutation**
  - **IP^{-1} – Inverse Permutation**
  - **PC1 – Permuted Choice 1**
  - **PC2 – Permuted Choice 2**
  - **E – Expansion (Permutation expands 32-bit values to 48 bits)**
  - **P – Post-S box primitive permutation**
  - **S-Boxes – Substitution Boxes**

- The first 64-bit input block is fed into the IP unit, which separates the block into a right half and a left half, each 32-bits wide after the permutation process.
- The **IP** is a mixing operation that is defined according to the following table:

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

- The above table simply indicates that bit 58 of the input sequence will become the 1st bit of the output sequence of the IP. In similar fashion, bit 50 becomes the 2nd bit, bit 2 the 8th bit and so on until bit 7, which becomes the last (64th) bit of the permuted output.

- The 64-bit thus permuted sequence is then divided into two 32-bit halves, the left (L) and the right (R) halves.

- At the end of the 16 rounds, the 64-bit block undergoes and **Inverse Permutation IP^{-1}** before producing the final 64-bit output. IP^{-1} reverses or cancels the effects of IP.

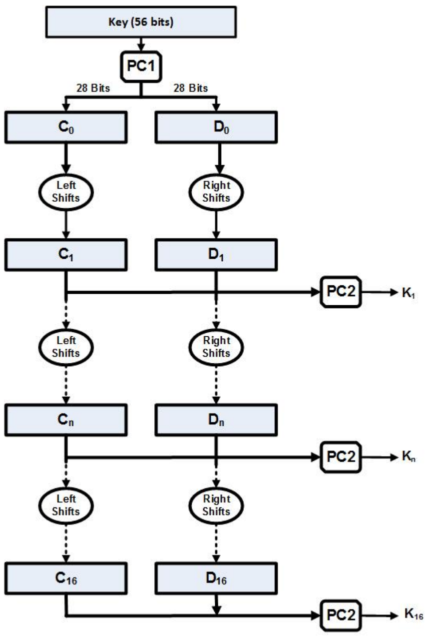
- The **IP**$^{-1}$ is a mixing operation that is defined according to the following table:

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**

- Just as before, the above table indicates that bit 40 of the input sequence will become the 1st bit of the output sequence of the IP, bit 8 becomes the 2nd bit, bit 32 the 8th bit and so on until bit 25, which becomes the last (64th) bit of the permuted output.

Key and Subkey Generation

- The diagram above provides an overview of the generation of the 16 subkeys from the original 64-bit key.
- The key generator distributes the original 56 bits of the key over the 16 subkeys. The following diagram illustrates the details of this process:



DES Subkey Generation

- The algorithm specifies a sequence of one or two circular shifts in the shift registers of the key generator to generate the 16 subkeys.
- Left shifts occur during encryption and right shifts occur during decryption.
- The resulting subkeys thus appear in the opposite order for these two processes.
- The first 28 bits out of PC1 is called $\mathbf{C_0}$, and the last 28 bits is $\mathbf{D_0}$. (Note that PC1 only uses 56 of the original 64 bits.)
- These are then left shifted according to a schedule until 16 subkeys are created. For instance, C_1 and D_1 are generated by left shifting C_0 and D_0 by one place, C_2 and D_2 by a further one place, and so on.

- **PC1** is a permuted choice that selects the 56 key bits from the original 64 bits, and then permutes them according to the following table :

C_0

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

D_0

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

- The table has two sections C_0 and D_0, each 28 bits. Note that the 8 parity bits in the key are not shown in the table.

- The original key bits are numbered from 1 to 64. Reading from the table above, the permuted bits of C_0 are respectively bits 57, 49, 41, etc., with the last (28th bit) being bit 36 of the original input.

- The permuted bits of D_0 are respectively bits 63, 55, 47, etc., with the last (28th bit) being bit 4 of the original input.

- The 16 subkeys are generated by successive shifts of the C and D segments according to the key schedule shown below:

Iteration Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- This procedure is followed by another permutation, PC2 (Permuted Choice 2), which is defined as follows:

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

- Using the above tables we can now summarize an example of a subkey generation as follows:
    - The 64-bit input key is first subjected to PC1, where the parity bits are stripped off, this reducing it to 56 bits and split into two 28-bit segments, C_0 and D_0.
    - Then as per the key schedule, one left shift is performed on C_0 and D_0.
    - The results from the above step are stored as C_1 and D_1 for the next cycle in the subkey generation, and are also subjected to PC2, thus forming the first subkey K_1.
    - As specified by the PC2 table, the 1st bit of K_1 is 14th bit of the concatenated output C_1D_1, the 2nd bit of K_1 is the 17th bit of C_1D_1, and so on, until the 48th bit of K_1, which is the 32nd bit of C_1D_1.
    - Next, as per the key schedule, one left shift is performed on C_1 and D_1, and the output is passed on to C_2 and D_2 for generating subkey K_2.
    - The concatenated output, C_2D_2 is subjected to PC2 to produce subkey K_2.

- The above process is repeated a total of 16 times; in each repetition the number of left shifts (one or two) is determined by the key schedule.

- For decryption, right shifts are performed instead of left shifts.

<u>Substitution Box Operations</u>

- Once the 16 subkeys have been generated, and the permutation IP has been performed on the 64-bit data block, the result is separated into two L and R blocks.

- Next, **substitution box** (S-Box) operations are performed on the contents of R and the contents of L are added modulo-2 to the results.

- The first step is to expand the contents of the R block from 32 bits to 48 bits.

- The **expansion function (E)** is the expansion function which takes a block of 32 bits as input and produces a block of 48 bits as output according to the following table:

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

- The table basically shows which output positions correspond to which input positions. For example, the 3rd bit position of the input block moves to the 4th bit position of the output block.

- Similarly, 21st bit position of the input block moves to the 30th and 32nd bit positions of the output block.

- Essentially this table divides the contents of the R block into 8 blocks of 6 bit positions each. It does this by adding the last two bit positions from the previous block to the beginning of each block.

- Thus, the last two bit positions of the first block are replicated at the start of the next block. For example, the last two bit positions of the first block, bits 4 and 5, are added to the front of the next block, and so on.

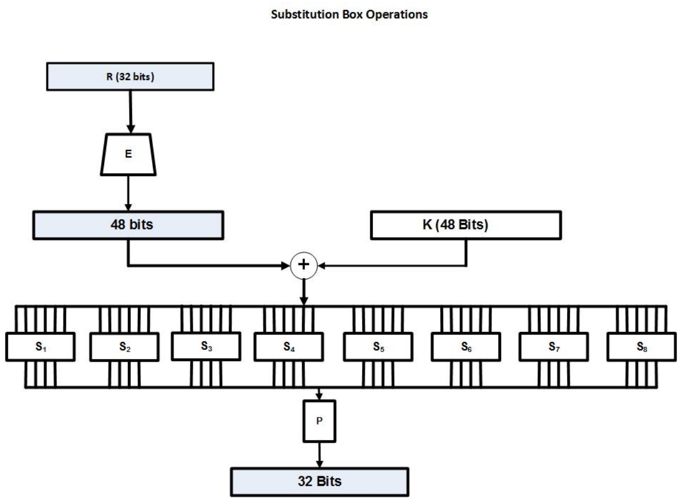
- Note that the last two bit positions for the very last block are wrapped around to the start of the very first block.

- The final result of this scheme is that it achieves in expanding the original 32 bits to a total of 48 bits using eight blocks of six bits.

- These 48 bits are then added modulo 2 to the first subkey K_1, which also 48 bits wide (56 bits – 8 parity bits) and the 48-bit result becomes the input to the S-boxes.

- Each of the 8 S-boxes has a 6-bit input and a 4-bit output. The 48 input bits are fed to the S-boxes in blocks of 6 bits.

- The following diagram provides and expanded view of this process:



- The procedure for mapping the **6 input bits** into **4 output bits** is outlined in the following tables:

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	8
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

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	4	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

- Although the mapping functions are completely defined, the technique used to derive them has never been revealed.
  - The output of the S-boxes is determined as follows:
  - The first and last bits of the 6-bit input block represent a base 2 number ***i*** in the range: 0 to 3.
  - The middle 4 bits of the input block represent a base 2 number ***j*** in the range: 0 to 15

- The ***i***th row and ***j***th column of the appropriate mapping table is a number in the range: 0 to 15, and is uniquely represented by a 4-bit block.

- Consider the following example:

  Input into S_1 (Bits 1 - 6) : **100110** (Bits 31 – 36 of XOR function)

  Bits 1 & 6: 10_2 = $\mathbf{2_{10}}$ => **Row 2**

  Bits 2 – 5 (middle 4 bits): 0011_2 = $\mathbf{3_{10}}$ => **Column 3**

  Thus, S_1 table entry (remember to count row and column indices from 0) = $\mathbf{8_{10}}$ => **1000** (the 4-bit output).

- The output of the S-box mapping is 8 blocks of 4 bits each, which are concatenated into 32 bits, with the S_1 output coming first and the S_8 output coming last.

- These 32 bits are then fed into the permutation **P** function, called the primitive function.

- The **P** function rearranges the order of the 32-bit result of the S-box operations as defined below:

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**

- This permutation maps each input bit position to an output bit position. No bits are used twice and no bits are ignored.

- For example the 16th bit of the input block becomes the 1st output bit; the 7th input bit becomes the 2nd output bit, and son until the 25th bit of the input block, which becomes the 32nd (and last) bit of the output block.

- Next, the 32-bit output of **P** is **XOR**'d with the 32-bit **L** block and the output is temporarily held in the **Temp** block.
- Then the contents of **R** are transferred to **L**, and the contents of **Temp** are fed into the R block (in other words a swap is executed).
- This last step concludes **one iteration** of the **16 iterations** required to perform a **single cycle** of the DES algorithm.
- The second iteration proceeds just as the first one, except that it starts with the results of the first (previous) iteration in the **L** and **R** blocks, and uses **subkey K_2**.
- The process continues until all **16 rounds**, each with a **different subkey** are concluded at the end of the algorithm.
- After all 16 rounds through the algorithm have been completed, the contents of **L** and **R** are concatenated in the opposite order, i.e., **R_1L_1, R_2L_2, ………$R_{32}L_{32}$**.
- Then the resulting 64-bit block is subjected to the inverse permutation IP^{-1} as discussed earlier.
- Finally, the output of IP-1 is either ciphertext or plaintext depending on what the original input was.
- At this point one complete cycle of the DES algorithm is concluded, and another 64-bit block of data is entered for encryption or decryption.
- Note (once again) that decryption uses the same algorithm as encryption, except that the subkeys K_1, K_2,…K_{16} are applied in **reversed order**.