Symmetric Block Ciphers - DES

Information Security – Lecture 05

Aadil Zia Khan

Symmetric-key Ciphers

- Symmetric-key ciphers use the same key for both the encryption of plaintext and the decryption of ciphertext
- Symmetric-key encryption is of two types
  - Stream ciphers encrypt the bytes of a message one at a time
  - Block ciphers take multiple bytes and encrypt them as a single unit
    - Plaintext can be padded so that it is a multiple of the block size

Achieving Confidentiality - Goal

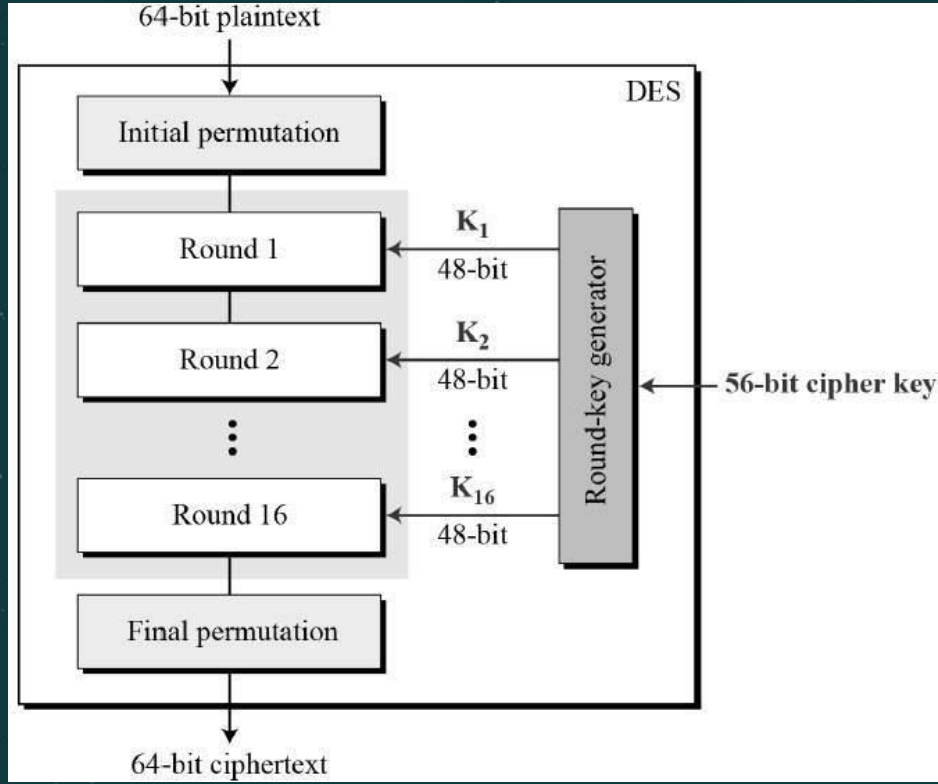
- Cipher needs to completely obscure statistical properties of original message - a one-time pad does this
  - E.g., you should not be able to identify the most frequent characters
  - E.g., you should not be able to infer the plaintext-ciphertext pairs

Achieving Confidentiality - Confusion & Diffusion

- How can a cipher obscure statistical properties of original message
  - Diffusion: spread out the statistical structure of plaintext over ciphertext
  - Confusion: relationship between ciphertext and key must be difficult to identify
- How exactly???
  - Shannon introduced idea of substitution-permutation (S-P) networks
  - Substitution (S-box): replace the bytes with other bytes
  - Permutation (transposition) (P-box): reorder the bytes

Data Encryption Standard (DES)

- Split the plaintext into 64bit blocks
- Use a 64bit key to generate a 56bit key which is used to further generate 48bit round subkeys
- Use the subkey to substitute and permute the plaintext
- Repeat 16 times - using a different subkey each time



DES - Subkey Generation

- Take the original 64bits of the key
- Permute according to the table
  - E.g., 57th bit of the original key becomes the first bit of the permuted key
  - E.g., 49th bit of the original key becomes the second bit of the permuted
- Note only 56 bits of the original key appear in the permuted key

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

DES - Subkey Generation

- Split the permuted key into right R_0 and left L_0 halves (28bits each)
- In $Round_1$ take R_0 and L_0, left shift according to the table, concatenate the two to get the $Subkey_1$ for that round
- In each round, take R and L of the previous round's subkey, left shift according to the table, concatenate the two to get the subkey for that round
- WAIT – one more step is needed to get the final subkey for each round

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

DES - Subkey Generation

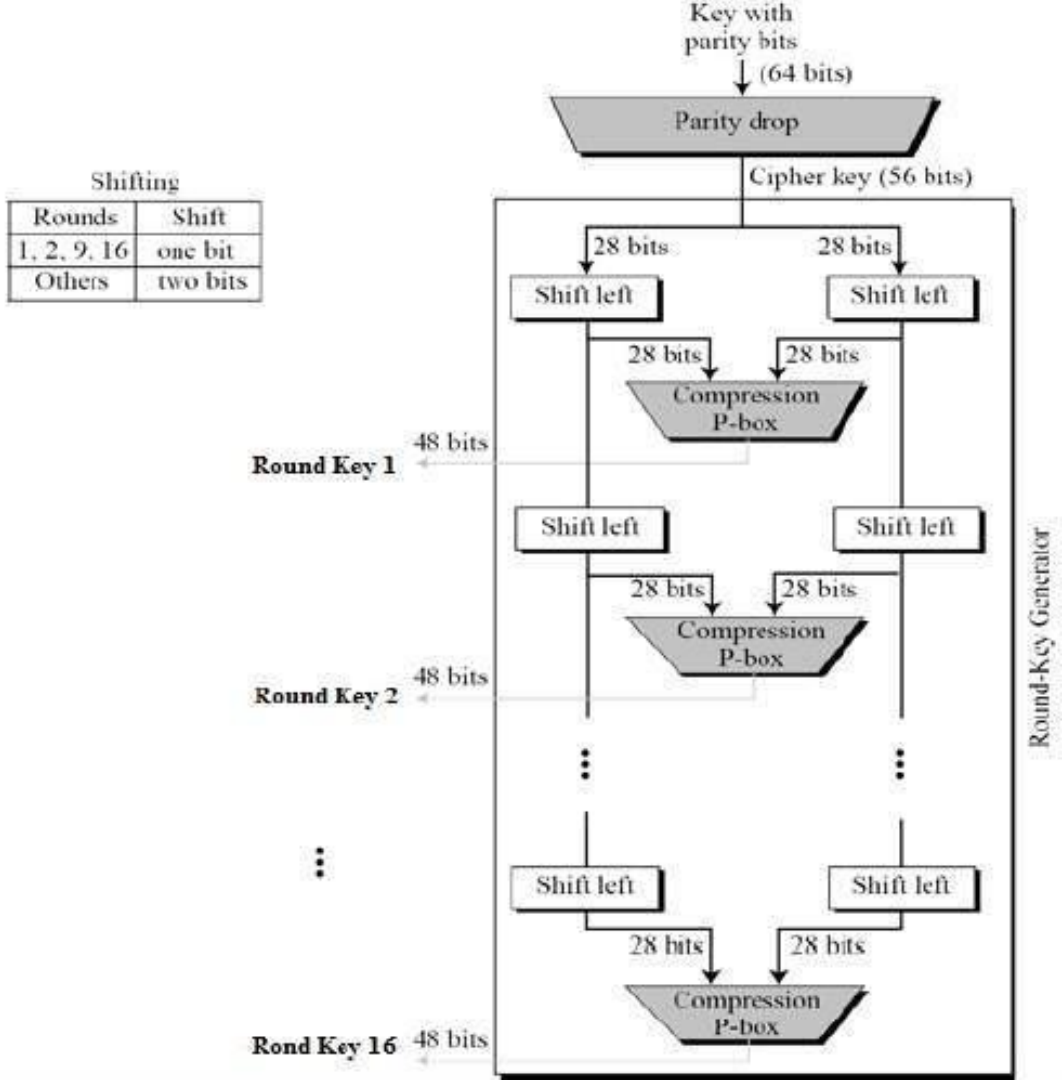
- Take the 56bit subkey for each round
- Permute according to the table
- Note only 48bits of the 56bit subkey appear in the permuted subkey

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

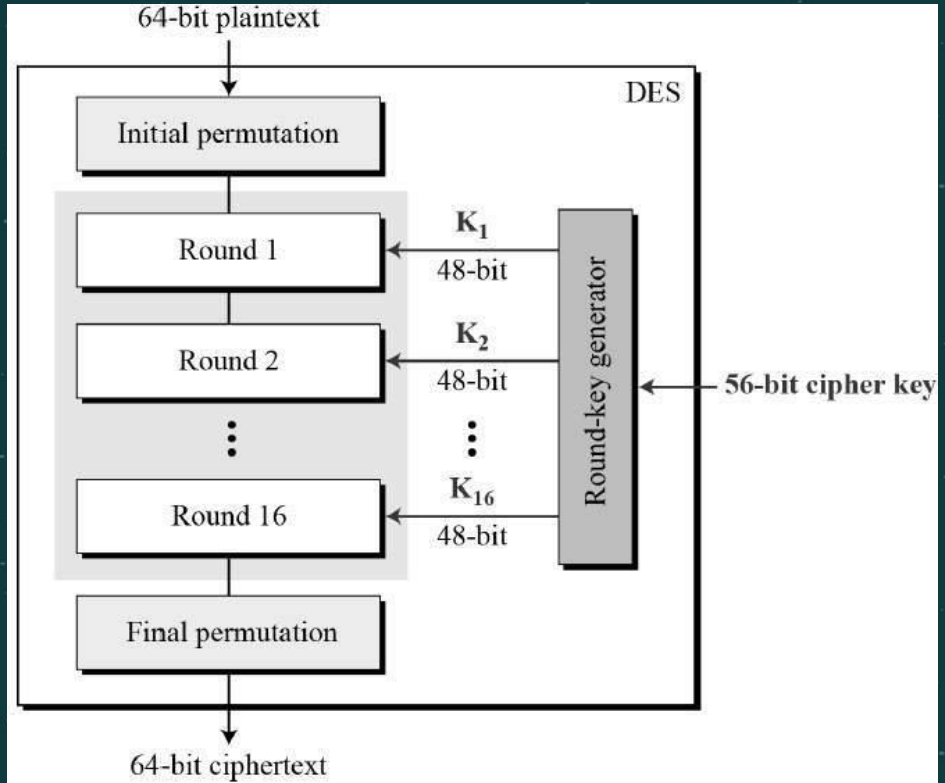
DES - Subkey Generation

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



DES Encryption



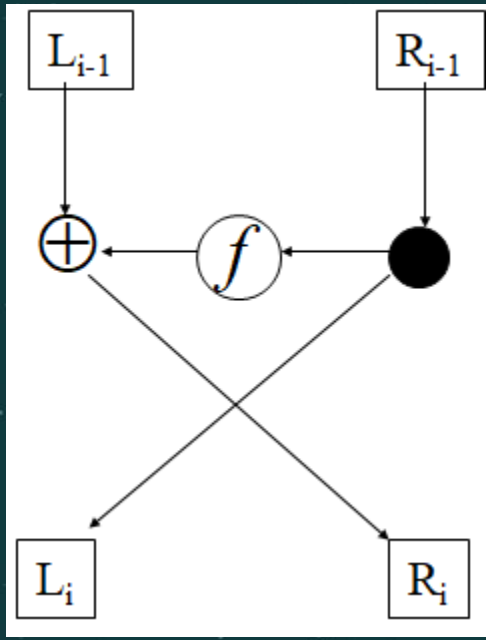
DES Encryption

- Split the plaintext into 64bit blocks
- Initially permute according to the table
  - At the end of the 16th round, you will once again permute using a table that is the inverse of this 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

DES Encryption - Round

- Divide the 64 bits into left, and right halves
- The right half becomes the left half in the next round
- Right half is also fed to the round function together with the key
- What happens to the left half?

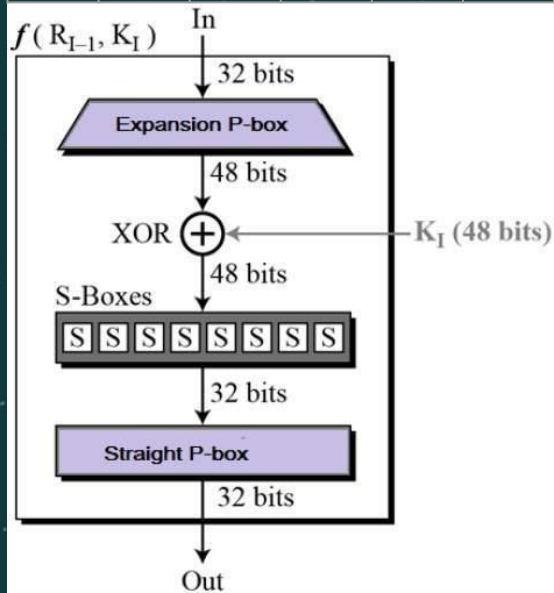


DES Encryption - Round

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

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

- Take the left half and permute/expand using the table – 32bits become 48bits
- XOR them with the 48bits subkey of that round
- Pass the resulting 48bits to 8 different substitution boxes
  - 6bits are passed to each Sbox, and 4bits are output
- Resulting 32bits are then permuted according to the table



DES - SBox

S1

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

- Lets take the example of S1 to see how Sboxes work – the same rules apply when using the other seven Sboxes
- The first and last bits of the input represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11) – this specifies the row
- The middle 4 bits of the input represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111) – this specifies the column
- E.g., for input 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence $S_1(011011) = 0101$.

DES Strength

- Is DES really strong?

Avalanche Effect

- A change of one input or key bit should result in changing half the output bits
  - This results in a strong cipher
- DES exhibits strong Avalanche Effect

DES Key Strength

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- It is now very easy to brute force a 56bits key

- If one machine is slow, we can distribute the task
  - We use multiple machines and each machine tries only a subset of all the keys
  - Since this is done in parallel, time is saved

Triple DES

- Triple-DES is just DES with two 56-bit keys applied
- Given a plaintext message, the first key is used to DES- encrypt the message
- The second key is used to DES-decrypt the encrypted message - this just scrambles the data further
- The twice-scrambled message is then encrypted again with the first key to yield the final ciphertext
- The resultant key space is about 2^112