Cryptography

Section 7

Created by/ Rehab Mohamed & Hanaa Mansour

Simplified Data Encryption Standard S-DES

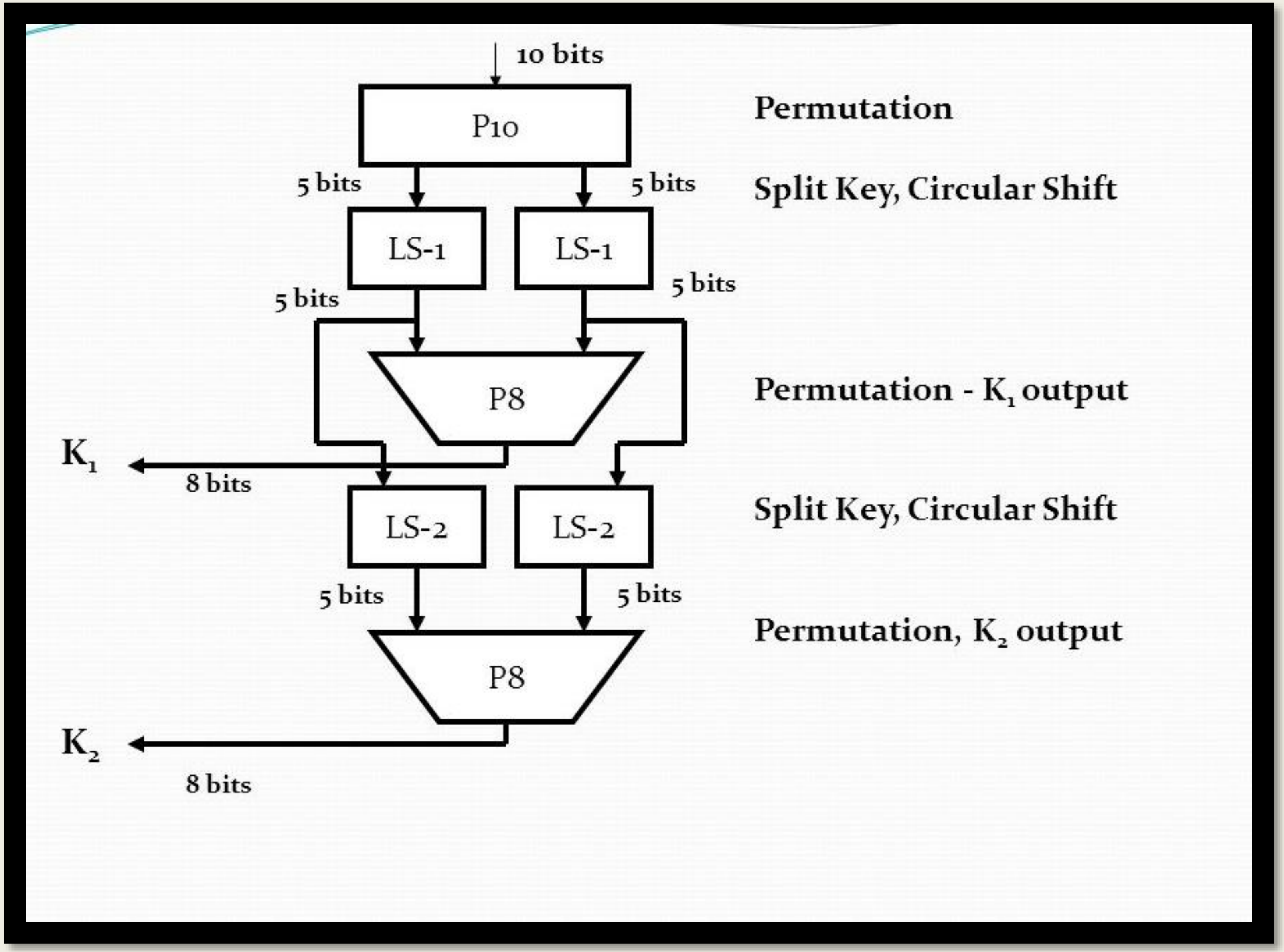
The overall structure of the simplified DES:

The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output.

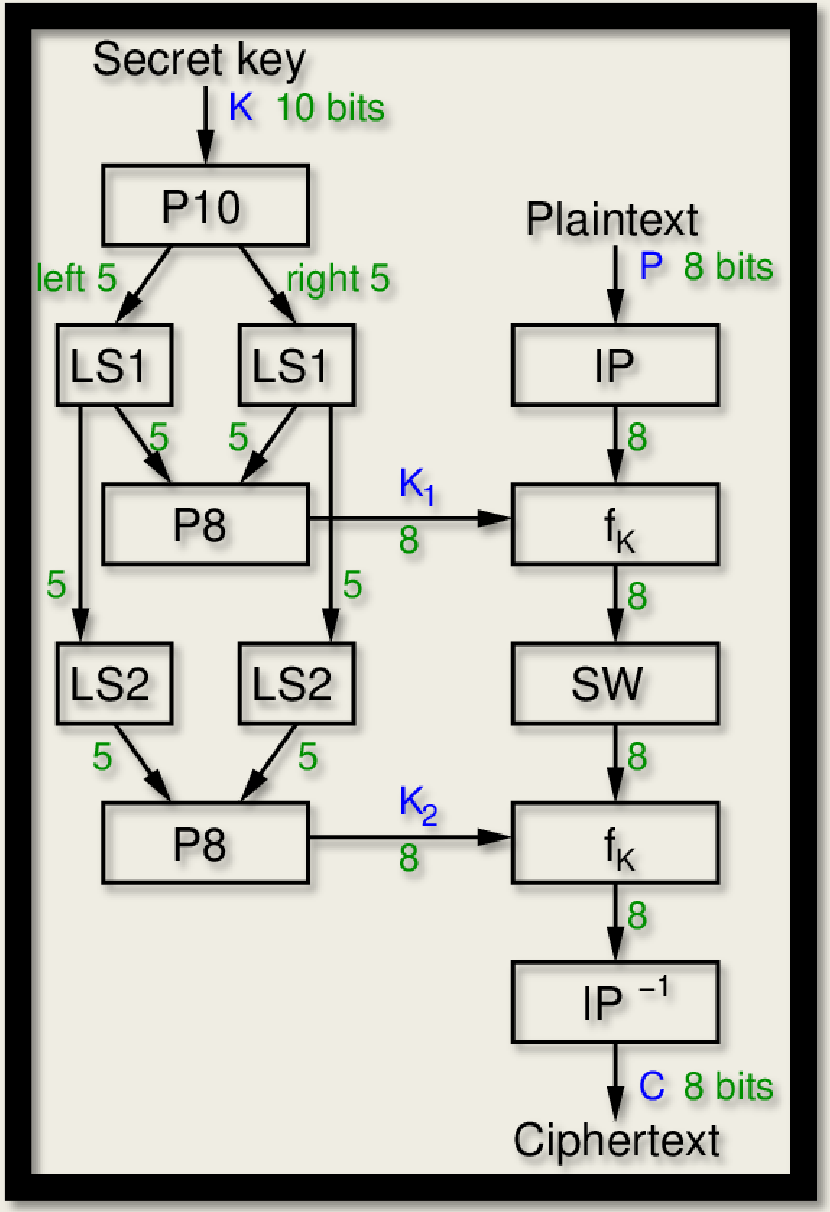
The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.

Simplified DES Key Generation

In the key generation algorithm, we accept the 10-bit key and convert it into two 8 bit keys. This key is shared between both sender and receiver.



Simplified DES Key Generation



Example on S-DES Key Generation

Given key **K = 1010000010, Find K1 and K2.**

Step 1: We accepted a 10-bit key and permuted the bits by putting them in the P10 table.

P10									
3	5	2	7	4	10	1	9	8	6

K_P10: 1000001100

Step 2:We divide the key into 2 halves of 5-bit each.

l = 10000 r = 01100

Step 3: Now we apply one bit left-shift on each key.

Kshifted: l = 00001 r =11000

Step 4: Combine both keys after step 3 and permute the bits by putting them in the P8 table. The output of the given table is the first key K1.

After LS-1 combined, we get 0 0 0 0 1 1 1 0 0 0

P8							
6	3	7	4	8	5	10	9

K_P8: 1 0 1 0 0 1 0 0 → **K1**

Step 5: The output obtained from step 3 i.e. 2 halves after one bit left shift should again undergo the process of two-bit left shift.

Step 3 output - l = 0 0 0 0 1, r = 1 1 0 0 0
After two bit shift - l = 0 0 1 0 0, r = 0 0 0 1 1

Step 6: Combine the 2 halves obtained from step 5 and permute them by putting them in the P8 table. The output of the given table is the second key K2.

After LS-2 combined = 0 0 1 0 0 0 0 0 1 1

P8							
6	3	7	4	8	5	10	9

After P8, we get Key-2 : 0 1 0 0 0 0 1 1 → K2

Final Output:

Key-1 is: 1 0 1 0 0 1 0 0

Key-2 is: 0 1 0 0 0 0 1 1

Key Generation Python Code

```
FIXED_P10 = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]
FIXED_P8 = [6, 3, 7, 4, 8, 5, 10, 9]
key = '1010000010’

def permutate(original, fixed_key):
    new = ''
    for i in fixed_key:
        new += original[i - 1]
    return new

def left_half(bits):
    return bits[:int(len(bits)) // 2]

def right_half(bits):
    return bits[int(len(bits)) // 2:]
```

```python
def shift(bits):
    rotated_left_half = left_half(bits)[1:] + left_half(bits)[0]
    rotated_right_half = right_half(bits)[1:] + right_half(bits)[0]
    return rotated_left_half + rotated_right_half



def key1():
    return permutate(shift(permutate(key, FIXED_P10)), FIXED_P8)



def key2():
    return permutate(shift(shift(shift(permutate(key, FIXED_P10)))), FIXED_P8)

k1 = key1()
k2 = key2()
print("The first key is " + k1)
print("The second key is " + k2)
```

<u>Output:</u>
The first key is 10100100
The second key is 01000011

Substitution-box (S-Box) in DES

b1 b2 b3 b4 b5 b6

1) Determine the row, by taking the first and last number from the given binary number, then convert to decimal.
2) Determine the column, by taking the middle four numbers between the first and last numbers from the given binary number, then convert to decimal.
3) Determine the row and column on the required box and take the intersection between row and column.
4) Convert the intersection number to binary (4 bit).

Remark: the input is 6 bits, and the output will be 4 bits

Example: The input to S-Box1 is 100011, what is the output

1 0 0 0 1 1

Step1: Determine the row

1 0 0 0 1 1

1 1 → 3

Step2: Determine the column

1 0 0 0 1 1

0 0 0 1 → 1

2 1

1 1

3 = 1+2

8 4 2 1

0 0 0 1

1 =

		Column															
Box	**Row**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S_1	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
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

So, we have (S-Box1 , Row = 3, column = 1)

Step 3: Determine the row and column on the required box and take the intersection between row and column.

		Column															
Box	Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	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

Step 4: Convert the intersection number to binary (4 bit).

Output is : 1100

8	4	2	1
1	1	0	0


Thank
You