الجامعة المستنصرية /كلية التربية / قسم علوم الحاسبات

4th Class
Computers & Data Security
أمنية الحاسوب والبيانات

أستاذ المادة

أ.م . د . اخلاص عباس البحراني

DES Example

- Let **M** be the plain text message **M** = 0123456789ABCDEF, where **M** is in hexadecimal (base 16) format. Rewriting **M** in binary format, we get the 64-bit block of text:
- **M** = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
  L = 0000 0001 0010 0011 0100 0101 0110 0111
  R = 1000 1001 1010 1011 1100 1101 1110 1111
- The first bit of **M** is "0". The last bit is "1". We read from left to right.
- Let **K** be the hexadecimal key **K** = 133457799BBCDFF1. This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):
- **K** = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001
- The DES algorithm uses the following steps:

1- **Step 1: Create 16 subkeys, each of which is 48-bits long:-** The 64-bit key is permuted according to the following table, **PC-1**. From the original 64-bit key

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

- Next, split this key into left and right halves, C_0 and D_0, where each half has 28 bits.

- C_o = 1111000 0110011 0010101 0101111
  D_o = 0101010 1011001 1001111 0001111
- With C_o and D_o defined, we now create sixteen blocks C_n and D_n, 1<=n<=16. Each pair of blocks C_n and D_n is formed from the previous pair C_{n-1} and D_{n-1}, respectively, for n = 1, 2, ..., 16, using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

Iteration Number	Number Left Shi
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

- This means, for example, C_3 and D_3 are obtained from C_2 and D_2, respectively, by two left shifts, and C_{16} and D_{16} are obtained from C_{15} and D_{15}, respectively, by one left shift.
- In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,…, 28, 1.
- From original pair pair C_0 and D_0 we obtain:
- C_0 = 1111000011001100101010101111
  D_0 = 0101010101100110011110001111
- C_1 = 1110000110011001010101011111
  D_1 = 1010101011001100111100011110
- C_2 = 1100001100110010101010111111
  D_2 = 0101010110011001111000111101
- .
- .
- C_{16} = 1111000011001100101010101111
  D_{16} = 0101010101100110011110001111
- We now form the keys K_n, for 1<=n<=16, by applying the PC-2 table to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but **PC-2** only uses 48 of these.
- For the first key we have C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

- which, after we apply the permutation **PC-2**, becomes
- K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

2- **Step 2: Encode each 64-bit block of data:-** Applying the initial permutation to the block of text **M**, given previously, we get

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Next divide the permuted block **IP** into a left half L_o of 32 bits, and a right half R_o of 32 bits.

L_o = 1100 1100 0000 0000 1100 1100 1111 1111
R_o = 1111 0000 1010 1010 1111 0000 1010 1010

- We now proceed through 16 iterations, for 1<=n<=16, using a function f which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, (bit-by-bit addition modulo 2)**. Then for **n** going from 1 to 16 we calculate
- $L_n = R_{n-1}$
  $R_n = L_{n-1} + f(R_{n-1}, K_n)$
- For n = 1, we have
- K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
  $L_1 = R_o$ = 1111 0000 1010 1010 1111 0000 1010 1010
  $R_1 = L_o + f(R_o, K_1)$

- We calculate $\mathbf{E}(R_o)$ from R_o as follows:
- R_o = 1111 0000 1010 1010 1111 0000 1010 1010
  $\mathbf{E}(R_o)$ = 011110 100001 010101 010101 011110 100001 010101 010101
- Next in the ***f*** calculation, we XOR the output $\mathbf{E}(R_{n-1})$ with the key K_n:
- $K_n + \mathbf{E}(R_{n-1})$.
- K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
  $\mathbf{E}(R_o)$ = 011110 100001 010101 010101 011110 100001 010101 010101
  $K_1 + \mathbf{E}(R_o)$ = 011000 010001 011110 111010 100001 100110 010100 100111.
- Write the previous result, which is 48 bits, in the form:
- $K_n + \mathbf{E}(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$, where each B_i is a group of six bits. We now calculate
- $S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$
- where $S_i(B_i)$ referres to the output of the ***i***-th **S** box.

- For the first round, we obtain as the output of the eight **S** boxes:
- $K_1 + E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.
- $S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111
- The final stage in the calculation of f is to do a permutation **P** of the **S**-box output to obtain the final value of f:
- $f = P(S_1(B_1)S_2(B_2)...S_8(B_8))$
- From the output of the eight **S** boxes, we get
- f = 0010 0011 0100 1010 1010 1001 1011 1011$R_1 = L_0 + f(R_0 , K_1)$

- = 1100 1100 0000 0000 1100 1100 1111 1111
  + 0010 0011 0100 1010 1010 1001 1011 1011
  = 1110 1111 0100 1010 0110 0101 0100 0100
- This is an example of one round od DES algorithm.