DES - Lecture #2

Friday, October 2, 2020 11:36 AM

DES Round Function

Output: 32 bits
Input Block: 32 bits
Round Key: 48 bits

32 bits 48 bits --> output will be 32 bits.
f(Input Block A, Round Key k) {

1. Calculate E(A), which is 48 bits. (E is an expansion matrix which works just like a permutation matrix.)
2. Calculate B = E(A) XOR k, XOR the round key with the expansion (48 bits)
3. Let $B = b_1 b_2 \ldots b_8$, subdivided into eight 6 bit blocks.
4. Calculate $c_i = S_i(b_i)$, for 1 <= i <= 8, The S's are called S boxes. There are 8 S boxes, each box takes in an input of 6 bits, and produces an output of 4 bits. These are hard-coded and non-linear. $C = c_1 c_2 \ldots c_8$ is the 32 bit output.
5. Return P(C), which is a permutation of the bits in C.

}

S-boxes

B = 011101 101011 000110 110110 … (4 more blocks of 6 bits

S1(b1) = S1(011101) = row 01, col 1110 0-based ro1 1, col 14
S1 entry in row 1 col 14 = 3 = 0011

S2(b2) = S2(101011) = row 11, col 0101 0-based row 3, col 5
S2 entry in row 3 col 5 = 15 = 1111

S3(b3) = S3(000110) = row 00, col 0011 0-based row 0, col 3
S3 entry in row 0 col 3 = 14 = 1110

S4(b4) = S4(110110) = row 10, col 1011 0-based row 2, col 11
S4 entry in row 2 col 11 = 14 = 1110

S-box Criteria
P0) Each row is a permutation of 0, 1, …, 15
P1) No S-box is a linear or affine function of its inputs.
P2) Changing 1 input bit to an S-box, causes at least two output bits to change.
P3) S(x) and S(x XOR 001100) these two outputs differ in at least 2 bits.
P4) For any input x, S(x) != S(x XOR 110000) and
S(x) != S(x XOR 110100) and
S(x) != S(x XOR 111000) and
S(x) != S(x XOR 111100)
P5) If we examine a single input bit to any S box and fix it, while toggling the other 5 bits, and then focus on any one of the four output bits, the distribution of 0s and 1s from the output bits can't differ by more than 6. (There has to be at least 13 0s and at least 13 1s out of the 32 outputs.)

x1xxxx (there are 32 bit strings of length 6 with 1 in the second position)

Now, calculate S(x1xxxx) for all 32 to these inputs and write down all 32 outputs

yyyZ

Now, for all 32 outputs, let's consider bit number 4. We should see at least 13 0s out of those 32 outputs and at least 13 1s.

After 16 rounds, we will swap L16 and R16 to get R16L16 and then we run IP^{-1}(R16L16) and that is the ciphertext.

Decryption, you can largely do the process backwards. (Run the rounds backwards with the round keys from the appropriate rounds.)

Key Schedule$_1$

Input Key is 56 bits, but is presented as 64 bits with 8 checksum bits.

Let's label the key bits k_1 through k_{64} and of these bits, the checksum bits are k_8, k_{16}, k_{24}, k_{32}, k_{40}, k_{48}, k_{56}, k_{64}.

Checksum is ODD PARITY for the row.

1011 0011, that last bit, k_8 has to be 1 so that there are an odd # of 1s on the row.
0111 1100,

1100 0001,
0000 0010,
0010 1111,
1111 1110,
0101 1110,
1101 1111

In most books they would list the key in HEX:

b3 7c c1 02 2f fe 5e df

Algorithm

Take the initial key and split it in half, calling the left half C_0 and the right half D_0. Note that our bit numbers will range all the way to 63 since we keep the bit numbers from the original numbering.

1) PC-1(C_0D_0)

2) For 16 rounds, do this:
   for i = 1 to 16:
   $C_i = LS_i(C_i)$
   $D_i = LS_i(D_i)$
   $K_i = PC\text{-}2(C_iD_i)$

Left-Shift is a cyclic left shift of the buffer, by either 1 or 2 bits. There is a table that tells you whether to do 1 bit or 2.

After PC-1 Step - bits we grab from original 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

Calculate C1 and D1 by cyclic left shift on both buffers round 1 - it's 1 bit

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 <u>57</u>

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 <u>63</u>

We will apply PC-2 to this entire buffer above. Here is PC-2:

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 round 1 key, starts with the 10th bit from the original key, followed by the 51st bit from the original key, followed by the 34th bit of the original key, followed by the 60th bit of the original key.

If I wanted to speed up my encryption decryption, I would pre-calculate all 16 round keys as permutation matrices of which bits to select from the original key. The Round 1 Key matrix would start: 10, 51, 34, 60, …