DES - Lecture #1

Wednesday, September 30, 2020 11:36 AM

DES - Data Encryption Standard

US Gov't solicited different public entities to submit symmetric block ciphers to become DES in the early 1970s.

Of these candidates, the one that was selected was a cipher created by Horst Feistel, who was an employee of IBM.

Idea behind a block cipher:

1) Break the plaintext into blocks, each of the same size number of bits.
2) Use a key, which is some fixed number of bits as well.
3) Output a block of ciphertext, typically the same length as the plaintext block.

DES block size = 64 bits
DES key size = 56 bits

Feistel introduced the idea of Rounds.

Each round will use a "different" key that is generated from the original key.

DES has 16 rounds.
So it has the round keys k1, k2, …, k16.
Each of the round keys is 48 bits.

If you know the key, you can undo all the steps…

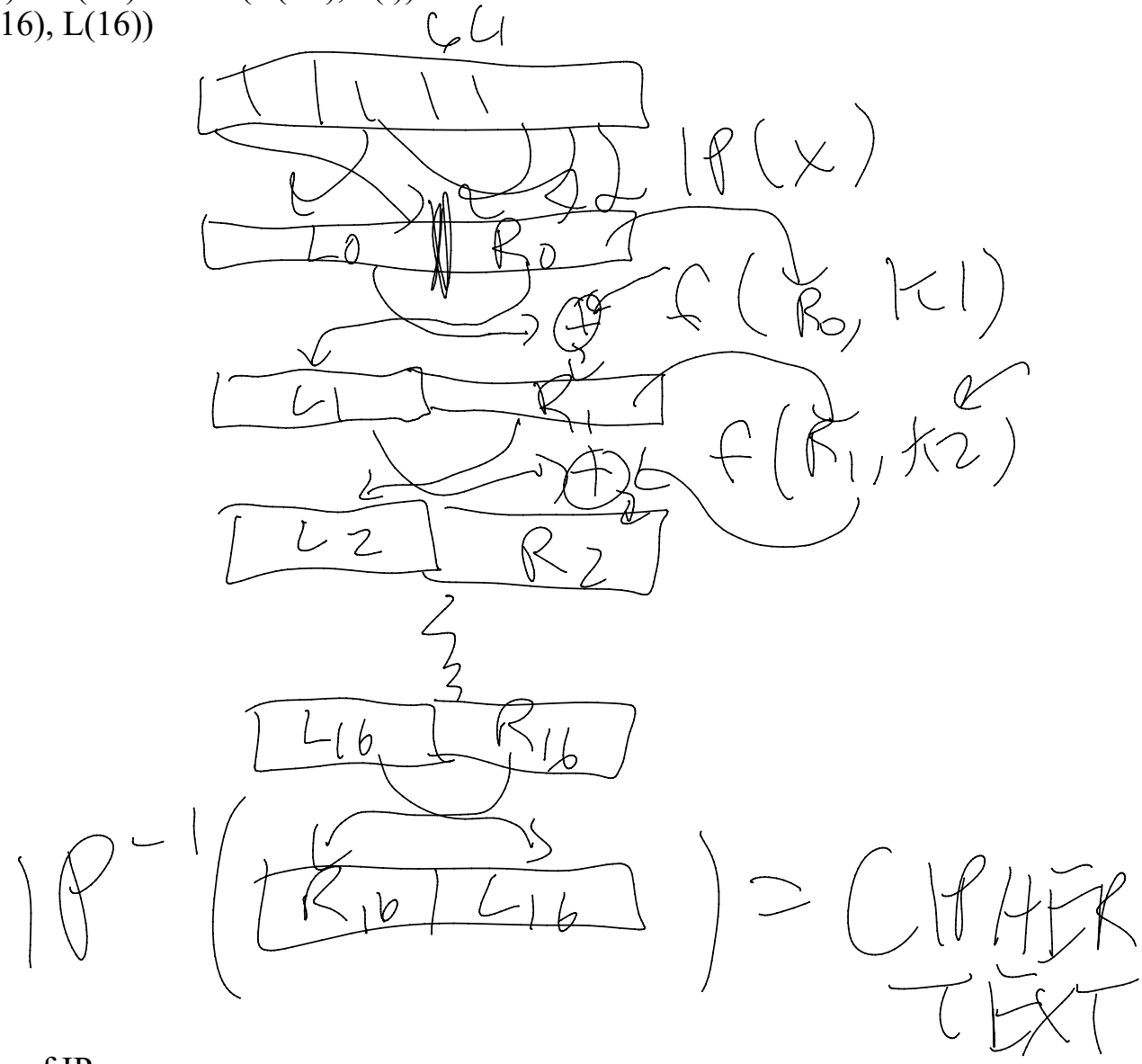
So, for right now, we will focus on simply encrypting a single block. Once we know how to encrypt a single block, then we can independently apply the cipher using different modes, which I will talk about later…

Input is a block x, key k

1. State = IP(x), IP stands for initial permutation.
   State = L0 (32 bits), R0 (32 bits)
2. for i=1 to 16:

a. L(i) = R(i-1)
b. R(i) = L(i-1) XOR f(R(i-1), k(i))

3. $IP^{-1}(R(16), L(16))$



Use of IP

Input block in Hex is AB34 F638 0942 5F52

1	0	1	0	1	0	1	1
0	0	1	1	0	1	0	0
1	1	1	1	0	1	1	0
0	0	1	1	1	0	0	0
0	0	0	0	1	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	0	1	0	0	1	0

So the first 8 bits are reading column 2 backwards (bits 58, 50, 42, etc.)

1110 0100 11001110 01000110 01010001

This is the first 32 bits of applying IP…just grab the bit number they tell you to and place it next.

Left as an exercise - find the last 32 bits of output.

The idea in a round is as follows:

L1 is just R0, so we are not messing with the bits of R0 at all in a round, but just moving them to the other half.

But, R1 is "messed up" compared to the previous inputs.

So rounds really go in pairs, with this structure, you would never have an odd number of rounds.

Now, let's focus on the function f:

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 $B = b_1b_2…b_8$, 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. $C = c_1c_2…c_8$ is the 32 bit output.
5. Return P(C), which is a permutation of the bits in C.

}

Example of E matrix:

Input: 8456 D13B

1	0	0	0	0	1	0	0
0	1	0	1	0	1	1	0
1	1	0	1	0	0	0	1
0	0	1	1	1	0	1	1

E tells me grab these bits: 32,1,2,3, 4,5,4,5, 6,7,8,9, 8,9,10,11, 12,13, 12,13,…

1100 0000 1000 0010 1010

Exercise: Complete the rest.

S-boxes

How to calculate S_1(010111) (in decimal this input equals 23…)

row = bit 1, bit 6 = 01, this means row 1
col = bit2,bit3,bit4,bit5 = 1011, this means column 11

Use 0 based indexing for rows and columns.

The entry in row 1, column 11 of S1 is 11, thus, $S_1(010111) = 11 = 1011_2$