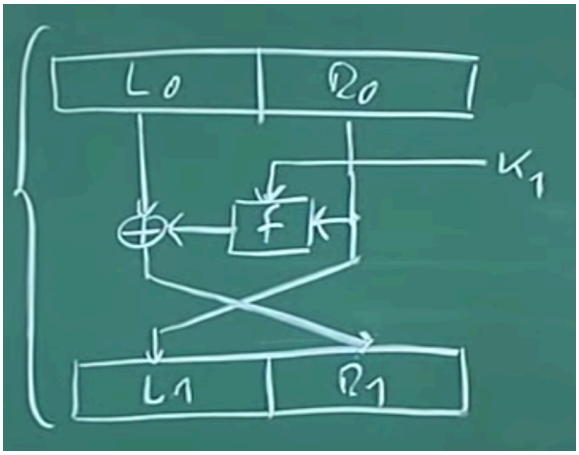
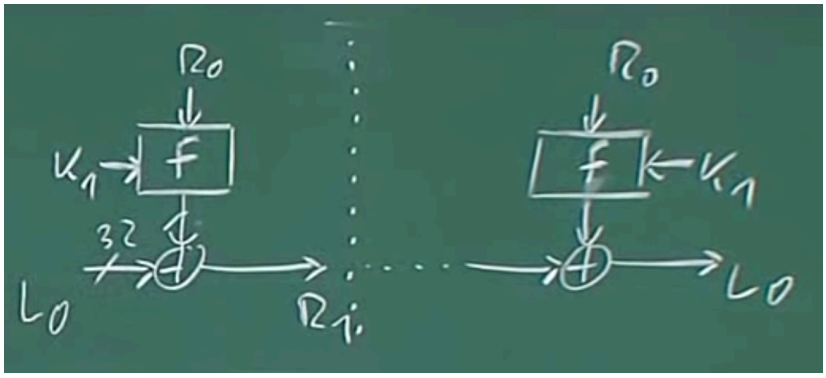
DES

- raw key length is 64 bits, but there are 8 parity bits, so the actual key length is 56 bits
- Symmetric cipher : use the same key for both encryption and decryption
- 16 rounds of identity operation , use different subkeys derived from the input main key
- meet the standard composed by Shannon
    - Confusion : Relationship between plain text and cipher should be obsecured. e.g. substitution
    - Diffusion : The influence of each plain text bit is spread over many cipher bits. e.g. permutation
- Combine Confusion and Diffusion many times to build a strong encryption algorithm
- Feistal scheme
    - split 64 bits into 2 parts, each with 32 bits
    - the right side become the left side of output
    - the left side of the output is `F_function(right side , subkey1) ^ left side`,and the `F_function` is very important, it is the heart of DES
    -



- the **left part** is the part which is encrypted, not the right part. Although we put the right part into the F_function, we paste it to the left side of output. The left part is encrypted with XOR, it take advantage of the result of F_function.



- The XOR part is just a process of Stream Cipher, in order to **ensure our data can be decrypted**, we must have the original R_0, so during the encryption, we paste it unchanged as L_1

- Why we have IP and IP^{-1}?
  - the permutation table is publicly known, it has nothing to do with security
  - It is there because this process facilitate the electronic implementation, it is a pure technical reason.
  - It is easy for hardware, but hard for software, so it can be ignored, maybe.

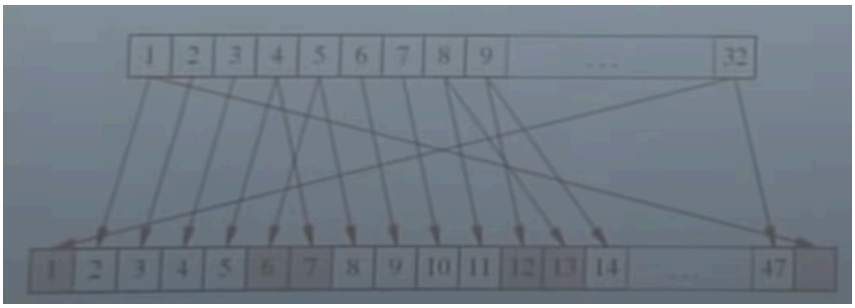
Details of the F_function

two input : $subK$ and R_0

four steps:

Step 1 : Expansion

- 32 bits in, 48 bits out
- provide Diffusion
- substitution actually



- some bits are mapped to 1 position each, but the left are mapped to 2 positions.

  Here is how diffusion comes, if 1 bit flips, it may influence 2 bits, then 4,8,16,.....

  And the number can be calculated by $48 - 32 = 16$

- here is the expansion table

E					
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

Step 2 : XOR

$$\text{result} \oplus \text{subkey}$$

Step 3 : Substitution box

the heart of heart

- 48 bits in, 32 bits out
- Consist of 8 s-boxes(means we have 8 substitution tables here), each one has 6 bits in, 4 bits out
- provide Confusion
- pick one of the S-box as example
  - 6 input , $2^6 = 64$ different bits pattern, decimal number from 0 to 63
  - so we have a table like

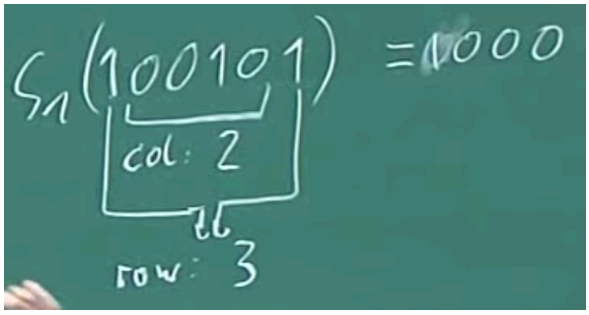
input	output
0b000000	0b1110(14)
.....	.....
0b111111	0b1101(13)

the mapping rule is **arbitrary!!!**, that is to say there is no rule between the input and output! (And the S-boxes are specially chosen to anti differential cryptanalysis , which is quite powerful in cracking modern cryptography)

And here is how we provide **Confusion**

- Another way to represent the table, 4 bits in the middle serve as row, left 2 serve as column

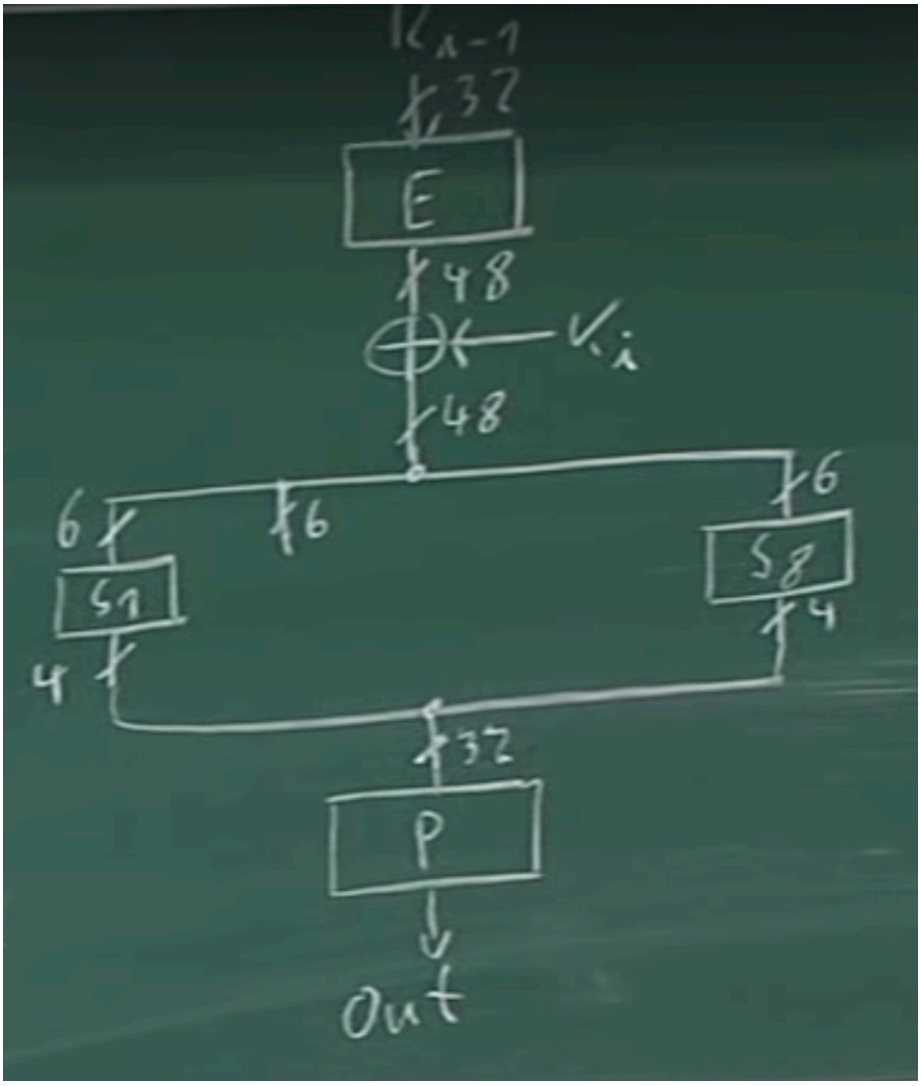
S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13



Step 4 : Permutation

only a simple permutation, distribute the fliped bits to random location, make it a global effect instead of a local one.

then we get the output of F_function



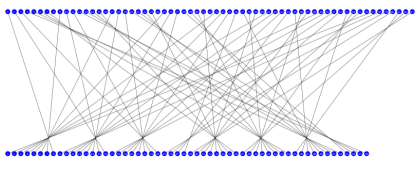
Key schedule

PC-1 (Permuted Choice 1)

- ignore the 8th, 16th, 24th....64th bits, that is to say drop the parity bits
- do the permutation with a fixed table, just like s-box, the table has no rule behind it.

PC-1

Left							*Right*						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4



`RecursionPoint:`

Split the bits into 28 bits + 28 bits

Left Shift

- It is a type of permutation
- both the left and right side should be shifted
- actually it is a rotation e.g. 1000 -> 0001

in round i , the length of bits should be shifted is that

$$\text{bits should be shifted} = \begin{cases} 1 & , i = 1, 2, 9, 16 \\ 2 & , i = \text{the rest of them} \end{cases}$$

it is interesting that, $k_0 = k_{15}$

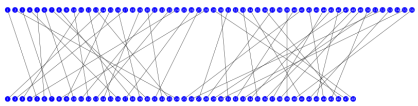
$$4 \times 1 + 2 \times 12 = 28$$

PC-2 (Permuted Choice 2)

- 56 bits in, 48 bits out, 8 bits are dropped
- again, the table is composed arbitrarily

Permuted choice 2 (PC-2) [edit]

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



This permutation selects the 48-bit subkey for each round from the 56-bit key-schedule state. This permutation will ignore 8 bits below:

Permuted Choice 2 "PC-2" Ignored bits 9,18,22,25,35,38,43,54.

then you get your sub key k_i

```
while(i!=15) goto RecursionPoint;
```

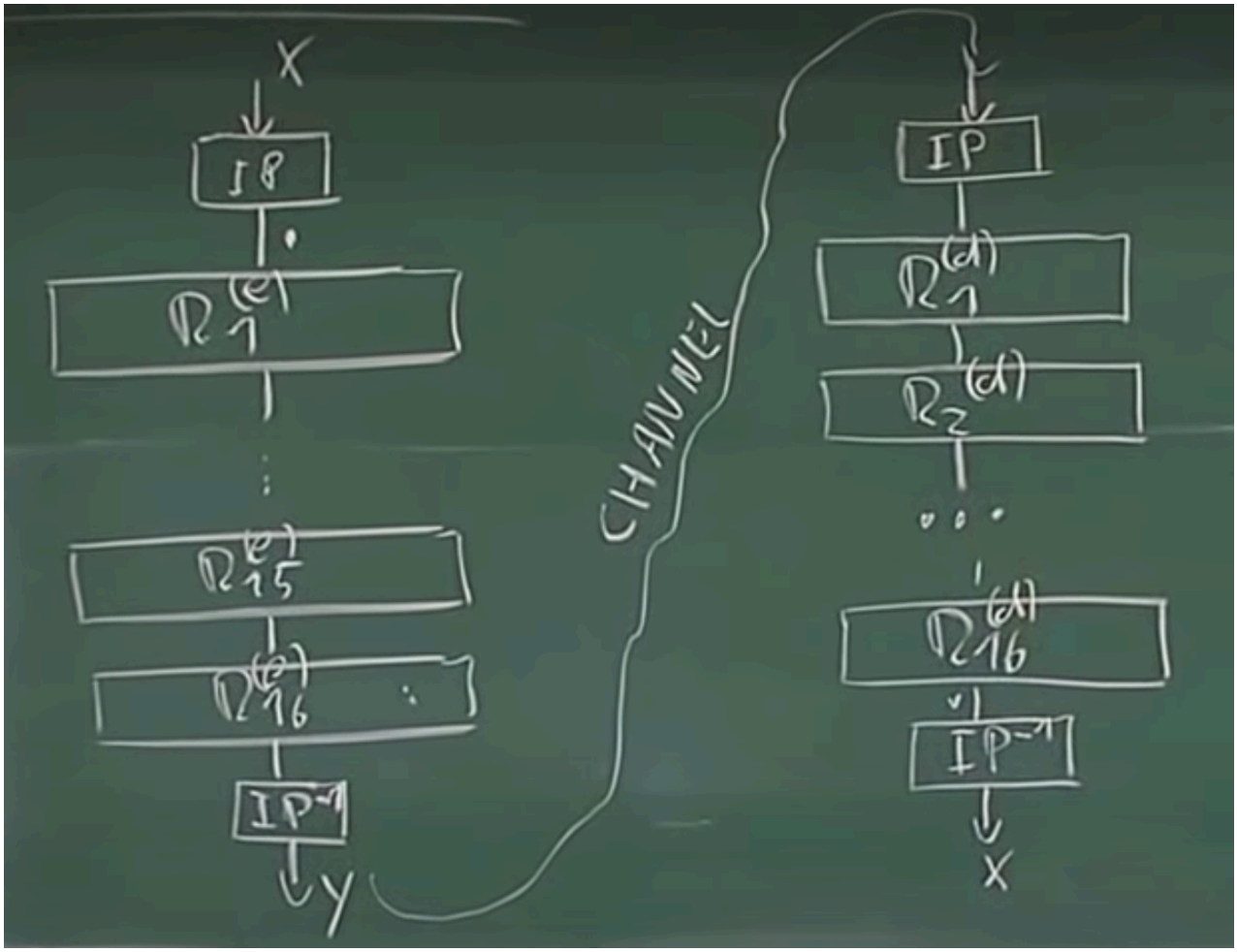
The sub keys are just permutation of the orginal prime key!!!

so why we don't merge the shift and the PC ?

we don't want 16 permutation tables, and this is more friendly to decryption process

---

Decryption



we only need to reverse one round because all rounds are identical.

recall feistal scheme, so we have

$$\begin{cases} L_0 = R_1 \oplus f(k_{16-i}, L_1) \\ R_0 = L_1 \end{cases}$$

when encryption, we don't swap left and right(or swap twice) in the 16th round in order to keep encryption algorithm identical with decryption algorithm

---

Python implementation from Github

video1

video2