# VE475 Introduction to Cryptography
# Homework 3

Jiang, Sifan
jasperrice@sjtu.edu.cn
515370910040

June 9, 2019

## Ex. 1 - Finite fields

1. Assume $X^2+1$ is reducible in $\mathbb{F}_3[X]$, then $X^2+1$ can be written as the product of two polynomials of lower degree. The possible factors of $X^2+1$ are $X$, $X+1$, and $X+2$.

$$\begin{aligned} X \cdot X &= X^2 \\ X \cdot (X+1) &= X^2 + X \\ X \cdot (X+2) &= X^2 + 2X \\ (X+1) \cdot (X+1) &= X^2 + 2X + 1 \\ (X+1) \cdot (X+2) &= X^2 + 2 \\ (X+2) \cdot (X+2) &= X^2 + X + 1 \end{aligned}$$

   Since none of them is equal to $X^2+1$, it is irreducible in $\mathbb{F}_3[X]$.

2. Since $X^2+1$ is irreducible in $\mathbb{F}_3[X]$ and $1+2X$ is a polynomial in $\mathbb{F}_3[X]$, there exists a polynomial $B(X)$ such that

$$(1+2X) \cdot B(X) \equiv 1 \mod (X^2+1)$$

   So, $B(X)$ is the multiplicative inverse of $1+2X \mod X^2+1$ in $F_3[X]$.

3. Using the extended Euclidean algorithm.

   Initially, $r_0 = X^2+1$, $r_1 = 2X+1$, $s_0 = 0$, $s_1 = 1$, $t_0 = 1$, and $t_1 = 0$.

| $q$ | $r_0$ | $r_1$ | $s_0$ | $s_1$ | $t_0$ | $t_1$ |
|---|---|---|---|---|---|---|
| 0 | $2X+1$ | $X^2+1$ | 1 | 0 | 0 | 1 |
| $2X$ | $X+1$ | $2X+1$ | $X$ | 1 | 1 | 0 |
| 2 | 2 | $X+1$ | $X+1$ | $X$ | 1 | 1 |
| $2X$ | 1 | 2 | $X^2+2X$ | $X+1$ | $X+1$ | 1 |
| 2 | 0 | 1 | $X^2+1$ | $X^2+2X$ | $X+2$ | $X+1$ |

   So, the multiplicative inverse of $1+2X \mod X^2+1$ in $\mathbb{F}_3[X]$ is $X^2+2X = 2+2X$.

## Ex. 2 - AES

1. (a) *InvShiftRows* should be described as: cyclically shift to the right row $i$ by offset $i$, $0 \leq i \leq 3$.

   (b) Since in the $AddRoundKey$ layer, each byte of the state is combined with a byte from the round key using the XOR operation ($\oplus$). Also, since $(a \oplus k) \oplus k = a$, where $a \in \{0, 1\}$, and $k \in \{0, 1\}$, the inverse of the layer $AddRoundKey$ is just applying $AddRoundKey$ again.

   (c) In *MixColumns* layer, we have $B = C(X) \times A$, where $B$ is the output matrix and $A$ is the state matrix. So the transformation *InvMixColumns* is given by $C^{-1}(X) \times C(X) \times A = A = C^{-1}(X) \times B$. We then need to verify if the result of the multiplication of the two matrices is an identity matrix or not.

$$\begin{aligned}
&\begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix} \\
\times &\begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix} \\
= &\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I
\end{aligned}$$

   The detailed calculation, take the element of first row and first column as the example, is

$$\begin{aligned}
I_{0,0} =&(00001110 \times 00000010) \oplus (00001011 \times 00000001) \oplus \\
&(00001101 \times 00000001) \oplus (00001001 \times 00000011) \\
=&00011100 \oplus 00001011 \oplus 00001101 \oplus (00010010 \oplus 00001001) \\
=&00011100 \oplus 00001011 \oplus 00001101 \oplus 00011011 \\
=&1
\end{aligned}$$

   So, the matrix is the inverse matrix of $C(X)$, and the transformation *InvMixColumns* is given by multiplication by the matrix.

2. In the first round, generate a round key according to the original 128 bits key. Then, apply the $AddRoundKey$ layer with columns $K(40), \cdots, K(43)$. And then apply *InvShiftRows* and *InvSubBytes* layers sequentially.

   In each round of the second to tenth rounds, sequentially apply layer $AddRoundKey$, *InvMixColumns*, *InvShiftRows*, and *InvSubBytes*. The columns used in $AddRoundKey$ layer in round $i$, where $i \in [2, 10]$, are $K(4(11-i)), \cdots, K(4(11-i)+3)$.

   In the last step, apply $AddRoundKey$ layer with columns $K_0, \cdots, K_3$.

3. Because *InvShiftRows* and *InvSubBytes* apply changes on each element "independently" and "linearly": *InvShiftRows* only shift the order of the elements but not changing the values; *InvSubBytes* map the state value to the combination of row and column numbers. So the order of applying these two layers won't change the result, thus can be applied on reverse order.

4. (a) Since $AddRoundKey$ apply operation element-wise while $InvMixColumns$ apply operation on the whole matrix, meaning they are using totally different methods and cannot be viewed as "linear". So, reversing the order of application of $AddRoundKey$ and $InvMixColumns$ would give different results.

   (b) The result of applying first $MixColumns$ and then $AddRoundKey$ is

   $$(a_{i,j}) \to (m_{i,j})(a_{i,j}) \oplus (k_{i,j})$$

   (c) Let $(e_{i,j}) = (m_{i,j})(a_{i,j}) \oplus (k_{i,j})$, we would have

   $$\begin{aligned}(e_{i,j}) &\to (m_{i,j})^{-1}(e_{i,j}) \oplus (m_{i,j})^{-1}(k_{i,j}) \\ &= (m_{i,j})^{-1}[(m_{i,j})(a_{i,j}) \oplus (k_{i,j})] \oplus (m_{i,j})^{-1}(k_{i,j}) \\ &= (m_{i,j})^{-1}(m_{i,j})(a_{i,j}) \oplus (m_{i,j})^{-1}(k_{i,j}) \oplus (m_{i,j})^{-1}(k_{i,j}) \\ &= (a_{i,j})\end{aligned}$$

   So the inverse operation is given by

   $$(e_{i,j}) \to (m_{i,j})^{-1}(e_{i,j}) \oplus (m_{i,j})^{-1}(k_{i,j})$$

   (d) First apply $InvMixColumns$ to the key and generate a new key, apply $AddRoundKey$ with the new key afterwards.

5. At the beginning, generate a round key according to the original $128$ bits key. Then, apply the $AddRoundKey$ layer with columns $K(40), \cdots, K(43)$.

   In each round of the first to ninth rounds, sequentially apply layer $InvSubBytes$, $InvShiftRows$, $InvMixColumns$, and $InvAddRoundKey$. The columns used in $InvAddRoundKey$ layer in round $i$, where $i \in [2, 10]$, are $K(4(11-i)), \cdots, K(4(11-i)+3)$.

   In the last step, apply apply layer $InvSubBytes$, $InvShiftRows$, and $AddRoundKey$ layer with columns $K_0, \cdots, K_3$ sequentially.

6. The advantage is that we don't need to change the order of the layers when decrypting the ciphertext, we only need to replace layers with their inverse procedure (except for two $AddRoundKey$ layers).

## Ex. 3 - DES

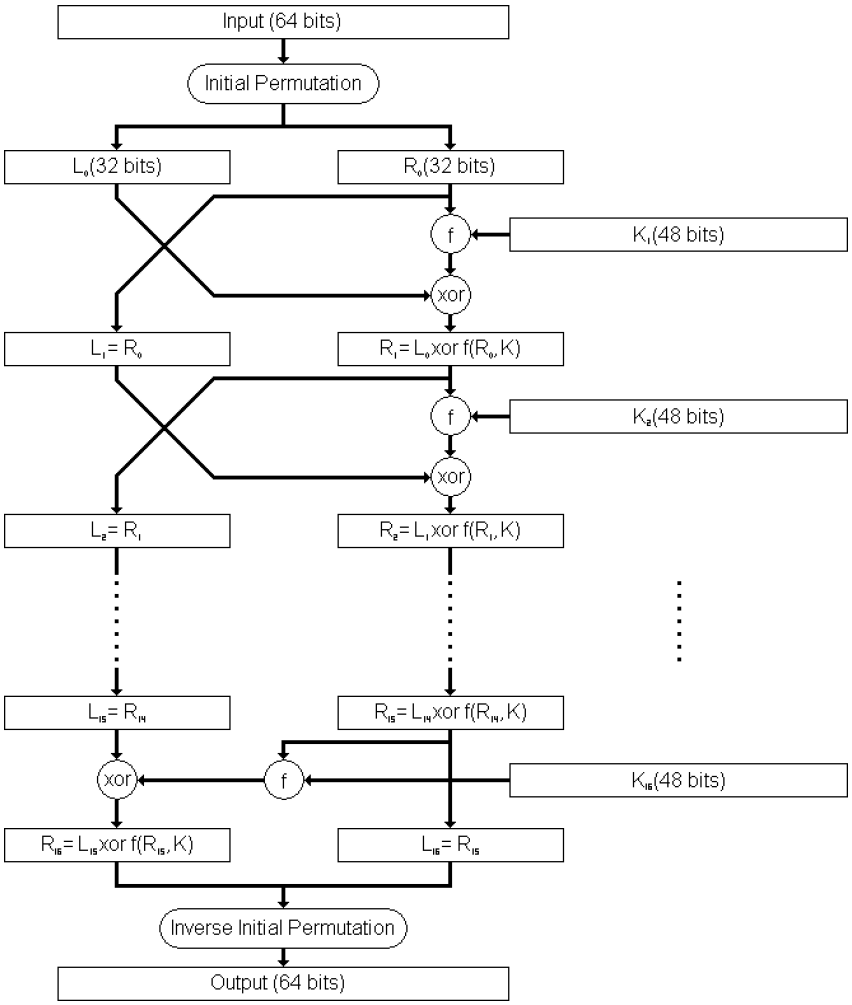


Figure 1: DES scheme.

1. The Data Encryption Standard is a symmetric-key block cipher and is an implementation of a Feistel Cipher. It uses 16 round functions with Feistel structure to encrypt the text. The block length is 64 bits, while the effective key length is 56 bits since 8 of 64 bits of the key are used to check parity. Besides the 16 rounds, there is also an initial and final permutation.

   The overall procedure of DES is shown in fig 1.

   - Initial and Final Permutation

     Initial permutation and final permutation are termed as $IP$ and $FP$ correspondingly. And they are inverses to each other.

     The initial permutation can be describe as: first rearrange the plaintext into an $8 \times 8$

matrix and shuffle the matrix according to the following order

$$\begin{bmatrix} 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & 48 \\ 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 \\ 57 & 58 & 59 & 60 & 61 & 62 & 63 & 64 \end{bmatrix} \xrightarrow{IP} \begin{bmatrix} 58 & 50 & 42 & 34 & 26 & 18 & 10 & 02 \\ 60 & 52 & 44 & 36 & 28 & 20 & 12 & 04 \\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 06 \\ 64 & 56 & 48 & 40 & 32 & 24 & 16 & 08 \\ 57 & 49 & 41 & 33 & 25 & 17 & 09 & 01 \\ 59 & 51 & 43 & 35 & 27 & 19 & 11 & 03 \\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 05 \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 & 07 \end{bmatrix}$$

The final permutation then follows the mapping

$$\begin{bmatrix} 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & 48 \\ 49 & 50 & 51 & 52 & 53 & 54 & 55 & 56 \\ 57 & 58 & 59 & 60 & 61 & 62 & 63 & 64 \end{bmatrix} \xrightarrow{FP} \begin{bmatrix} 40 & 08 & 48 & 16 & 56 & 24 & 64 & 32 \\ 39 & 07 & 47 & 15 & 55 & 23 & 63 & 31 \\ 38 & 06 & 46 & 14 & 54 & 22 & 62 & 30 \\ 37 & 05 & 45 & 13 & 53 & 21 & 61 & 29 \\ 36 & 04 & 44 & 12 & 52 & 20 & 60 & 28 \\ 35 & 03 & 43 & 11 & 51 & 19 & 59 & 27 \\ 34 & 02 & 42 & 10 & 50 & 18 & 58 & 26 \\ 33 & 01 & 41 & 09 & 49 & 17 & 57 & 25 \end{bmatrix}$$

- Round Function ("f" on fig 1)

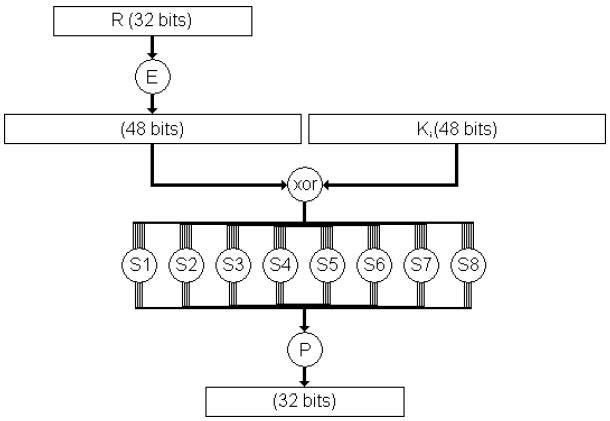


Figure 2: Round function.

The procedure of round function of $R_{i-1}$ and $K_i$ follows fig 2.

– Expansion Permutation ("E" on fig 2)
Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic can be described as

$$\begin{bmatrix} 01 & 02 & 03 & 04 \\ 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 \end{bmatrix} \xrightarrow{E} \left[\begin{array}{c|cccc|c} 32 & 01 & 02 & 03 & 04 & 05 \\ 04 & 05 & 06 & 07 & 08 & 09 \\ 08 & 09 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 01 \end{array}\right]$$

Where the extra digits are added on the left and right of the origin matrix.

– Substitution Boxes ("S1" to "S8" on fig 2)
Each block will perform a substitution with S-Box, so that map an 6-bit text $A = (a_1a_2a_3a_4a_5a_6)$ to a 4-bit text $B = (b_1b_2b_3b_4)$. We use $r = (a_1a_6)$ as the row number and $c = (a_2a_3a_4a_5)$ as the column number and then can find the value of $B$ according to the substitution table which is illustrated in tab 1.

Table 1: Substitution table.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
| $S_2$ | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |
| $S_3$ | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |
| $S_4$ | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |
| $S_5$ | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
| $S_6$ | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
| $S_7$ | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
| $S_8$ | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

– Straight Permutation ("P" on fig 2)
The 32-bit output of S-boxes is then subjected to the straight permutation with rule permutation logic

$$\begin{bmatrix} 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \end{bmatrix} \xrightarrow{P} \begin{bmatrix} 16 & 07 & 20 & 21 & 29 & 12 & 28 & 17 \\ 01 & 15 & 23 & 26 & 05 & 18 & 31 & 10 \\ 02 & 08 & 24 & 14 & 32 & 27 & 03 & 09 \\ 19 & 13 & 30 & 06 & 22 & 11 & 04 & 25 \end{bmatrix}$$

- Key Generation
Take the 64-bit key as input, the round-key generator creates sixteen 48-bit keys, corresponding to 16 round functions, out of a 56-bit cipher key. The procedure

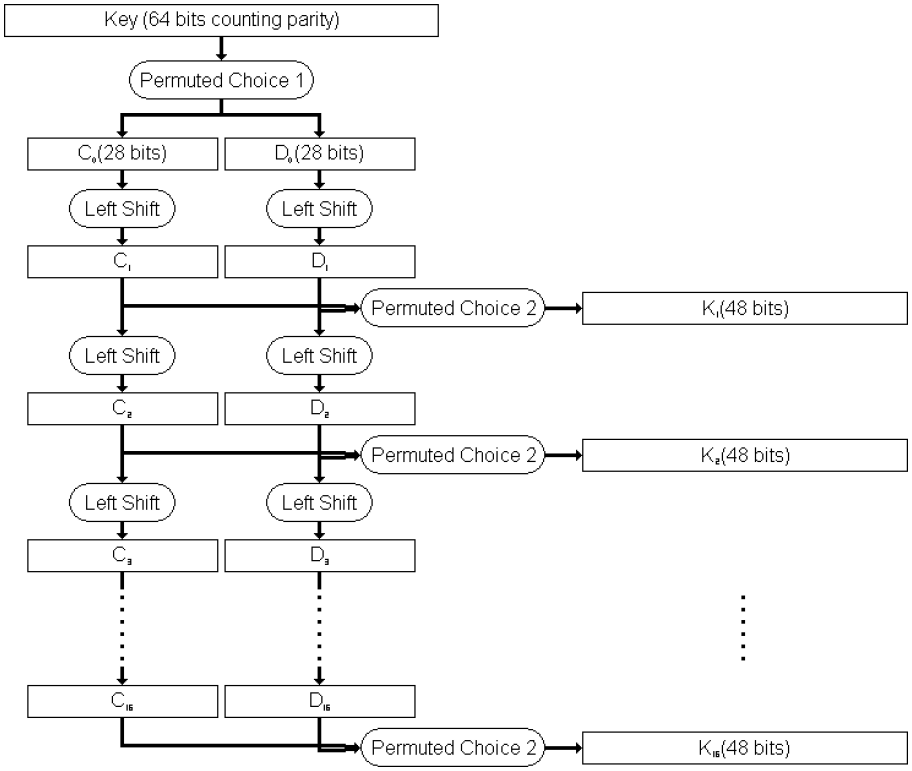


Figure 3: Key generation.

of key generation is shown graphically in fig 3. Every eighth bit is ignored and produces 56 bits.

– Permutation Choice 1
Apply PC-1 on the 56 bits, and the permutation logic is

$$\begin{bmatrix} 01 & 02 & 03 & 04 & 05 & 06 & 07 \\ 09 & 10 & 11 & 12 & 13 & 14 & 15 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 33 & 34 & 35 & 36 & 37 & 38 & 39 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 49 & 50 & 51 & 52 & 53 & 54 & 55 \\ 57 & 58 & 59 & 60 & 61 & 62 & 63 \end{bmatrix} \xrightarrow{PC-1} \begin{bmatrix} 57 & 49 & 41 & 33 & 25 & 17 & 09 \\ 01 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 02 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 03 & 60 & 52 & 44 & 36 \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 07 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 06 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 05 & 28 & 20 & 12 & 04 \end{bmatrix}$$

– The output from PC-1 is separated into the first two 28 bits: $C_0$ for left part and $D_0$ for right part.
– At each round, a circular left shift is performed on $C_{i-1}$ and $D_{i-1}$, the bits rotated follows tab 2.

Table 2: Bits rotated in "Left Shift".

| Round number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bits rotated | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

– Permutation Choice 2
The $C_{i-1}$ and $D_{i-1}$ in each round would pass through permutation choice two

to produce 48-bit key. The permutation logic is

$$\begin{bmatrix} 01 & 02 & 03 & 04 & 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & 48 \end{bmatrix} \xrightarrow{PC-2} \begin{bmatrix} 14 & 17 & 11 & 24 & 01 & 05 & 03 & 28 \\ 15 & 06 & 21 & 10 & 23 & 19 & 12 & 04 \\ 26 & 08 & 16 & 07 & 27 & 20 & 13 & 02 \\ 41 & 52 & 31 & 37 & 47 & 55 & 30 & 40 \\ 51 & 45 & 33 & 48 & 44 & 49 & 39 & 56 \\ 34 & 53 & 46 & 42 & 50 & 36 & 29 & 32 \end{bmatrix}$$

2. Linear and differential cryptanalysis are two most widely used attacks on block ciphers.

   Linear cryptanalysis: Linear cryptanalysis is a known plaintext attack (KPA) in which the attacker studies probabilistic linear relations (called linear approximations) between parity bits of the plaintext, the ciphertext, and the secret key. Given an approximation with high probability, the attacker obtains an estimate for the parity bit of the secret key by analysing the parity bits of the known plaintexts and ciphertexts. Using auxiliary techniques he can usually extend the attack to find more bits of the secret key.

   Differential cryptanalysis: the method searches for plaintext, ciphertext pairs whose difference is constant, and investigates the differential behaviour of the cryptosystem by finding the highest probability differential input which can be traced through several rounds. And then assign probabilities to the keys and locate the most probable key.

3. Triple DES applies the DES cipher algorithm three times to each data block with different keys. 3DES is used instead of double DES is because 2DES is vulnerable to meet-in-the-middle attack: given a known plaintext pair $(x, y)$, such that $y = K_2(K_1(x))$, Eve can break the keys in $2^n$ steps, where $n$ is the length of the key length which is 58 and cannot be viewed as secure.

4. According to "Hints for user passwords": The legacy UNIX System encryption method is based on the NBS DES algorithm. User account information is saved in /etc/passwd. Secure user account information is saved in /etc/shadow. It is no longer secure to encrypt data with DES.

## Ex. 4 - Programming