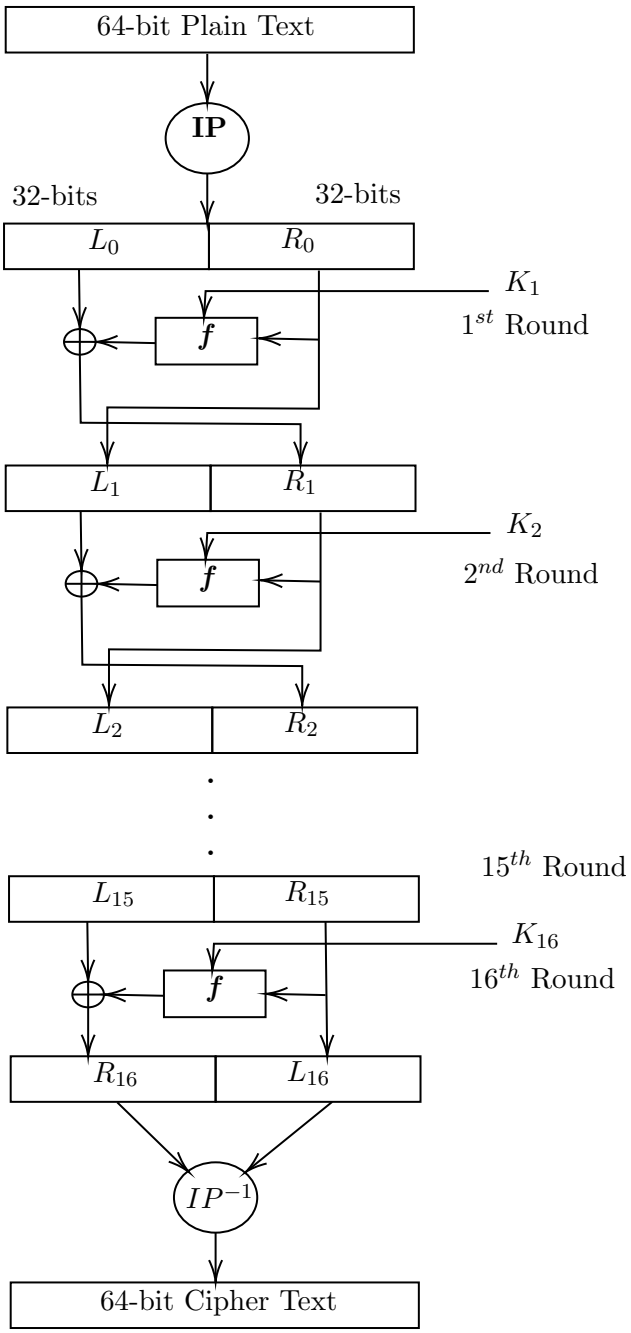
[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024

Scribed by: Manas Jitendrakumar Ingle (202151086) — Lecture 1,2 (Weak 1)

Data Encryption Standard

Flowchart for encrypting block of text using DES:



Steps in DES:

1. Initial Permutation (IP): The plaintext input is subjected to an initial permutation (IP) to rearrange the bits. This permutation is a fixed table that determines the new order of the bits.

2. Inverse Initial Permutation (IP^{-1}): Similar to the initial permutation, the inverse initial permutation is applied to the output after the last round. It essentially undoes the effect of the initial permutation and is used to obtain the final ciphertext.

3. Algorithm of f: The core of the DES algorithm is the function f, which operates on half of the data (32 bits) and involves the use of the round key. This function includes operations like expansion, XOR with the round key, substitution (using S-boxes), and permutation.

4. Key Scheduling Algorithm: DES uses a key scheduling algorithm to generate 16 subkeys, one for each round. The 56- bit key is transformed through various operations, including permutation and rotation, to produce these round keys.

Initial Permutation:

It is a bijection from 64-bit to 64-bit. The 64-bit message is permuted using the IP and then further encryption is done. Initial Permutation is defined as:

$$\text{IP} = \begin{bmatrix} 58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 \\ 60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 \\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 \\ 64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 \\ 57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 \\ 59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 \\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 \\ 63 & 55 & 47 & 39 & 31 & 23 & 25 & 7 \end{bmatrix}$$

The Initial Permutation can be understood as follows:

$$IP(m_1 m_2 \dots m_{64}) = m_{58} m_{50} m_{42} \dots m_7$$

Here, $m_1 m_2 \dots m_{64}$ represents the 64-bit plaintext, and $m_{58} m_{50} m_{42} \dots m_7$ represents the permuted output.We can easily compute its inverse and it will be equal to 8x8 matrix.
The permutation pattern for IP^{-1}in an 8x8 matrix:

$$IP^{-1} = \begin{bmatrix} 40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 \\ 39 & 7 & 47 & 15 & 55 & 23 & 63 & 31 \\ 38 & 6 & 46 & 14 & 54 & 22 & 62 & 30 \\ 37 & 5 & 45 & 13 & 53 & 21 & 61 & 29 \\ 36 & 4 & 44 & 12 & 52 & 20 & 60 & 28 \\ 35 & 3 & 43 & 11 & 51 & 19 & 59 & 27 \\ 34 & 2 & 42 & 10 & 50 & 18 & 58 & 26 \\ 33 & 1 & 41 & 9 & 49 & 17 & 57 & 25 \end{bmatrix}$$

Algorithm of f

The function $f(R_i, K_i) = X_{i+1}$, where:

$$f : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$$
$$f(R_i, K_i) = X_i$$
where,
R_i is 32-bit
K_i is 48-bit
X_{i+1} is 32-bit.

The round function for DES is defined as:

$$f(R_i, K_i) = P(S(E(R_i) \oplus K_i))$$

where,
Expansion Function E : $\{0,1\}^{32} \rightarrow \{0,1\}^{48}$
Substitution Box S: $\{0,1\}^{48} \rightarrow \{0,1\}^{32}$
Permutation Box P: $\{0,1\}^{32} \rightarrow \{0,1\}^{32}$
Hence,

length of R_i = 32-bits
length of $E(R_i)$ = 48-bits = length of K_i
length of $E(R_i) \oplus K_i$ = 48-bits
length of $S(E(R_i) \oplus K_i)$ = 32-bits
length of $P(S(E(R_i) \oplus K_i))$ = 32-bits

Expansion Function

$$E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$$

The expansion function for DES is given below:

$$E = \begin{bmatrix} 32 & 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 1 \end{bmatrix}$$

The bits are repeated for expanding 32-bits to 48-bits. In simple words, for each set of 4 bits, we add the LSB of prev set in the beginning of current set and the MSB of next set at the end of current set. We do in a circular manner for the first and last set.

$$E(x_1 x_2 .... x_{32}) = (x_{32} x_1 x_2 x_3 x_4 x_5 x_4 x_5 .... x_{32} x_1)$$

Substitution Box

$$S : \{0,1\}^{48} \rightarrow \{0,1\}^{32}$$
$$S(X) = Y, \text{ where X is 48 and Y is 32 bit long}$$

Dividing X into 8 parts each of length 6-bits.

$$X = B_1B_2B_3B_4B_5B_6B_7B_8$$

Corresponding to each B_i there is a substitution box S_i where $i \in \{1, 2, .., 8\}$.

$$S_i : \{0,1\}^6 \rightarrow \{0,1\}^4 \forall i \in \{1, 2, .., 8\}$$
$$S_i(B_i) = C_i$$
$$\therefore S(X) = (S_1(B_1), S_2(B_2), S_3(B_3), S_4(B_4), S_5(B_5), S_6(B_6), S_7(B_7), S_8(B_8))$$

Therefore, length of S(X) is 32 bits. The substitution boxes are given on page 260 of the book Ḧandbook of Applied Cryptography*:* Now let us see how to perform the conversion using Substitution box.

$$B_i = b_1b_2b_3b_4b_5b_6 \text{ , where } b_i \in \{0,1\}$$

We can find the row and column of the substitution box using these bits.

$$\text{r(row)} = 2 * b_1 + b_6,$$
$$\text{where r is integer representation of } b_1b_6 \text{ and } 0 \leq r \leq 3$$

$$\text{c(column)} = \text{integer representation of } b_2b_3b_4b_5$$
$$\text{where } 0 \leq c \leq 15$$

$$S_i = \begin{bmatrix} a_{0,0} & \dots & a_{0,15} \\ \vdots & \ddots & \vdots \\ a_{3,0} & \dots & a_{3,15} \end{bmatrix} \text{ where } a_{i,j} \in \{0, 1, ...15\}$$

now using this S_i :

$$S_i(B_i) = a_{r,c}$$

Permutation Box

$$P : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$$

It is also defined by a table. The table is given below:

$$P = \begin{bmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{bmatrix}$$

Using this, we can see the permutation as:

$$P(x_1x_2x_3x_4x_5...x_{32}) = x_{16}x_7x_{20}x_{21}x_{29}...x_{25}$$

Key Scheduling Algorithm

Input : 64-bit secret key
Output : 16 round keys where len(k_i) = 48 bits
Algorithm :

- Define $v_i, 1 \leq i \leq 16$, where $v_i = 1$ if $i \in \{1, 2, 9, 16\}$, else $v_i = 2$.
- Discard 8 parity check bits from K. The 56 bit key is $\tilde{\text{K}}$.
- T = PC1($\tilde{\text{K}}$), where PC1 is a permuation defined as:

$$PC1 : \{0,1\}^{56} \rightarrow \{0,1\}^{56}$$

- $(C_0, D_0) = T$, where C_0 is most significant 28 bits of T and D_0 is least significant 28 bits of T.
- for $i = 1$ to 16:

$$C_i = (C_{i-1} \hookleftarrow v_i)$$
$$D_i = (D_{i-1} \hookleftarrow v_i)$$
where $\hookleftarrow$ is left circular shift
For example: $x_1x_2x_3....x_28 \hookleftarrow 2 = x_3x_4x_5....x_1x_2$ $K_i = PC2(C_i, D_i)$

where, PC2 is a substitution defined as:

$$PC2 : \{0,1\}^{56} \rightarrow \{0,1\}^{48}$$

PC1(Permuted Choice 1)

It permutes the 56 bits of secret key before generating round keys.

$$PC1 : \{0,1\}^{56} \rightarrow \{0,1\}^{56}$$

For C_i:

$$\text{PC1} = \begin{bmatrix} 57 & 49 & 41 & 33 & 25 & 17 & 9 \\ 1 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 2 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 3 & 60 & 52 & 44 & 36 \end{bmatrix}$$

For D_i:

$$\text{PC1} = \begin{bmatrix} 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 7 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 6 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 5 & 28 & 20 & 12 & 4 \end{bmatrix}$$

$$PC1(k_1k_2 \ldots k_7k_9 \ldots k_{63}) = (k_{57}k_{49}k_{41}k_{33} \ldots k_9k_1k_{58} \ldots k_{63}k_{55} \ldots k_4)$$

PC2(Permuted Choice 2)

It is used to select 48 bits from the concatenation $b_1 b_2 \dots b_{56}$ of C_i and D_i

$$\text{PC2} = \begin{bmatrix} 14 & 17 & 11 & 24 & 1 & 5 \\ 3 & 28 & 15 & 6 & 21 & 10 \\ 23 & 19 & 12 & 4 & 26 & 8 \\ 16 & 7 & 27 & 20 & 13 & 2 \\ 41 & 52 & 31 & 37 & 47 & 55 \\ 30 & 40 & 51 & 45 & 33 & 48 \\ 44 & 49 & 39 & 56 & 34 & 53 \\ 46 & 42 & 50 & 36 & 29 & 32 \end{bmatrix}$$

$$K_i = b_{14} b_{17} b_{11} \dots b_3 2$$

Attack Models

1) **Ciphertext only attack:**
   - **Scenario:** Attacker is getting only ciphertext.
   - **Goal:** To get back the plaintext or recover the secret key.
2) **Known plaintext attack:**
   - **Scenario:** Attacker knows some plaintexts and corresponding ciphertexts.
   - **Goal:** Find a plaintext corresponding to a different ciphertext or find the secret key of the system.
3) **Chosen plaintext attack:**
   - **Scenario:** Attacker chooses some plaintexts of his/her choice and he/she is allowed to get the corresponding ciphertext.
   - **Goal:** Generate a new plaintext-ciphertext pair or find the secret key.
4) **Chosen ciphertext attack:**
   - **Scenario:** Attacker chooses some ciphertexts, and he/she is provided with the corresponding plaintexts.
   - **Goal:** Generate a different valid plaintext-ciphertext pair or find the secret key.

Chosen Plaintext Attack

- We perform exhaustive search for DES (56-bit). So the total set of possible keys: $k_1 k_2 k_3 \dots k_{56}$
  1. $M \to \text{DES}(M, K) = C_1$
  2. $M \to \text{DES}(M, K) = C_2$

  $\text{DES}(M, K) = C_2$
- Attacker: $\text{DES}(M, K_i) = \tilde{C}_i$ $\tilde{C}_i \neq C_1 \to K_i \neq K$ $\tilde{C}_i \neq C_2 \to K_i \neq K$ Thus we can delete both K_i and K_i from our searching set which reduces the set to half. Thus our new set becomes of size 2^{55} instead of 2^{56}.

Double DES:

Attacker is having one valid Plain text cipher text pair
$C \rightarrow DoubleDES$.
Selectk_i
$ENC_{DES}(P, K_i) = X_i$ {Table 1}
$ENC_{DES}(C, K_i) = Y_i$ {Table 2}
If $X_i = Y_i$, then we get to know that k_i, k_j is the secret key.

Triple DES

Triple DES Encryption

1. $M \xrightarrow{\text{DES}(M,K_1)} C_1$
2. $C_1 \xrightarrow{\text{DES}^{-1}(C_1,K_2)} M'$
3. $M' \xrightarrow{\text{DES}(M',K_1)} C_2$

To decrypt, the process is reversed:

Triple DES Decryption

1. $C_2 \xrightarrow{\text{DES}^{-1}(C_2,K_1)} M'$
2. $M' \xrightarrow{\text{DES}(M',K_2)} C_1$
3. $C_1 \xrightarrow{\text{DES}(C_1,K_1)} M$

Group

A group(G,*) consist of a set G with a binary operations * on G satisfying three axioms
1.Group operation is associative
a*(b*c) = (a*b)*c $\forall$ a,b,c $\in$ G

2.There is an element I $\in$ G called Identity element such that.
a*I = I*a = a $\forall$ a $\in$ G

3. For each a $\in$ G there exists an element $a^{-1} \in$ G
$a * a^{-1} = 1 = a^{-1} * a \forall a \in G$

4. Group G is commutative
$a * b = b * a \forall a, b \in G$

Group Examples

Example 1: $G = \{$**set of all invertible** $n \times n$ **matrices**$\}$ Define the group operation as matrix multiplication:

1. $A \times B \neq B \times A$ (Non-commutative)

2. $A * B \neq B * A$
3. $A * (B * C) = (A * B) * C$
4. $A * I_n = I_n * A = A$
5. $A * A^{-1} = A^{-1} * A = I_n$

Example 2: Z (Set of Integers) a. $(Z, +)$

1. $a + (b + c) = (a + b) + c$ for all $a, b, c \in Z$
2. 0 is the identity element: $a + 0 = 0 + a = a$
3. Inverse element: $a + (-a) = (-a) + a = 0$, thus $a^{-1} = -a$

b. $(Z, \times)$

1. $a \times (b \times c) = (a \times b) \times c$
2. $a \times 1 = 1 \times a = a$
3. No inverse element: There is no $a^{-1} \in Z$ such that $a * a^{-1} = 1$

c. $(Z, -)$

1. $a - (b - c) \neq (a - b) - c$ for all $a, b, c \in Z$
2. 0 is the identity element: $a - 0 \neq 0 - a$
3. $a - (-a) \neq (-a) - a$

Thus, $(Z, -)$ is not a valid group.

Quantum Search Algorithm's and Cryptography

I was fortunate to have done design project on Introduction to Quantum Computing which gave me exposure to various Quantum Algorithms like Grovers Search Algorithm and Shor's Algorithm . Grover's Search Algorithm con brute-force a 128 bit symmetric cryptographic key in roughly 2^{64} iterations. Complexity of Grover's Search Algorithm is $O(\sqrt{N})$.
By efficiently factoring large integers, Shor's Algorithm undermines the security of widely used encryption schemes like RSA, which rely on the difficulty of factoring large numbers for their security.