

DES Documentation

Made with Love, by Mousa Emarah

1. Key Generation Process:

1.1 Initial Permutation (PC1):

- The 64-bit key undergoes permutation using the PC1 table
- Results in a 56-bit key

1.2 Key Division:

- The 56-bit key is divided into two equal halves:
 - Left half (C_0): 28 bits
 - Right half (D_0): 28 bits

1.3 Subkey Creation:

- Sixteen subkeys are generated through 16 rounds
- Each round applies circular left shifts to both halves
- Shift amounts follow the predefined shift schedule
- Each shifted pair forms a 56-bit intermediate key

1.4 Final Subkey Permutation (PC2):

- Each 56-bit intermediate key is permuted using PC2 table
- Produces 16 final 48-bit subkeys (K_1 to K_{16})
- These subkeys are used in the Feistel network

2. Message Encryption Process:

2.1 Input Conversion:

- Plaintext message is converted to 64-bit binary representation
- Padding applied if message is not 64 bits

2.2 Initial Permutation (IP):

- 64-bit block undergoes permutation using IP table
- Reorders the bits according to IP table specification

2.3 Block Division:

- Permuted block is split into:
 - Left half (L_0): 32 bits
 - Right half (R_0): 32 bits

2.4 Feistel Rounds (16 iterations):

For each round n (1 to 16):

$$2.4.1 L_n = R_{n-1}$$

$$2.4.2 R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

Where:

- \oplus = XOR operation
- f = Feistel function
- K_n = Subkey for round n

2.5 Expansion (E):

- Right half (R_{n-1}) expanded from 32 to 48 bits
- Uses expansion table E
- Allows combination with 48-bit subkey

2.6 S-box Substitution:

- 48-bit result divided into eight 6-bit groups
- Each group processed through corresponding S-box (S1-S8)
- Each S-box outputs 4 bits
- Total output: 32 bits (8×4 bits)

2.7 P Permutation:

- 32-bit S-box output undergoes permutation
- Uses P table to rearrange bits
- Produces final output of Feistel function

2.8 XOR Operation:

- P-permuted result is XORed with L_{n-1}
- Produces new right half R_n
- Left half L_n becomes previous R_{n-1}

3. Final Steps:

3.1 After 16 rounds:

- Left and right halves are concatenated
- Note: No swap after final round

3.2 Final Permutation (FP):

- Combined block undergoes final permutation
- Uses inverse of initial permutation table
- Produces final 64-bit ciphertext

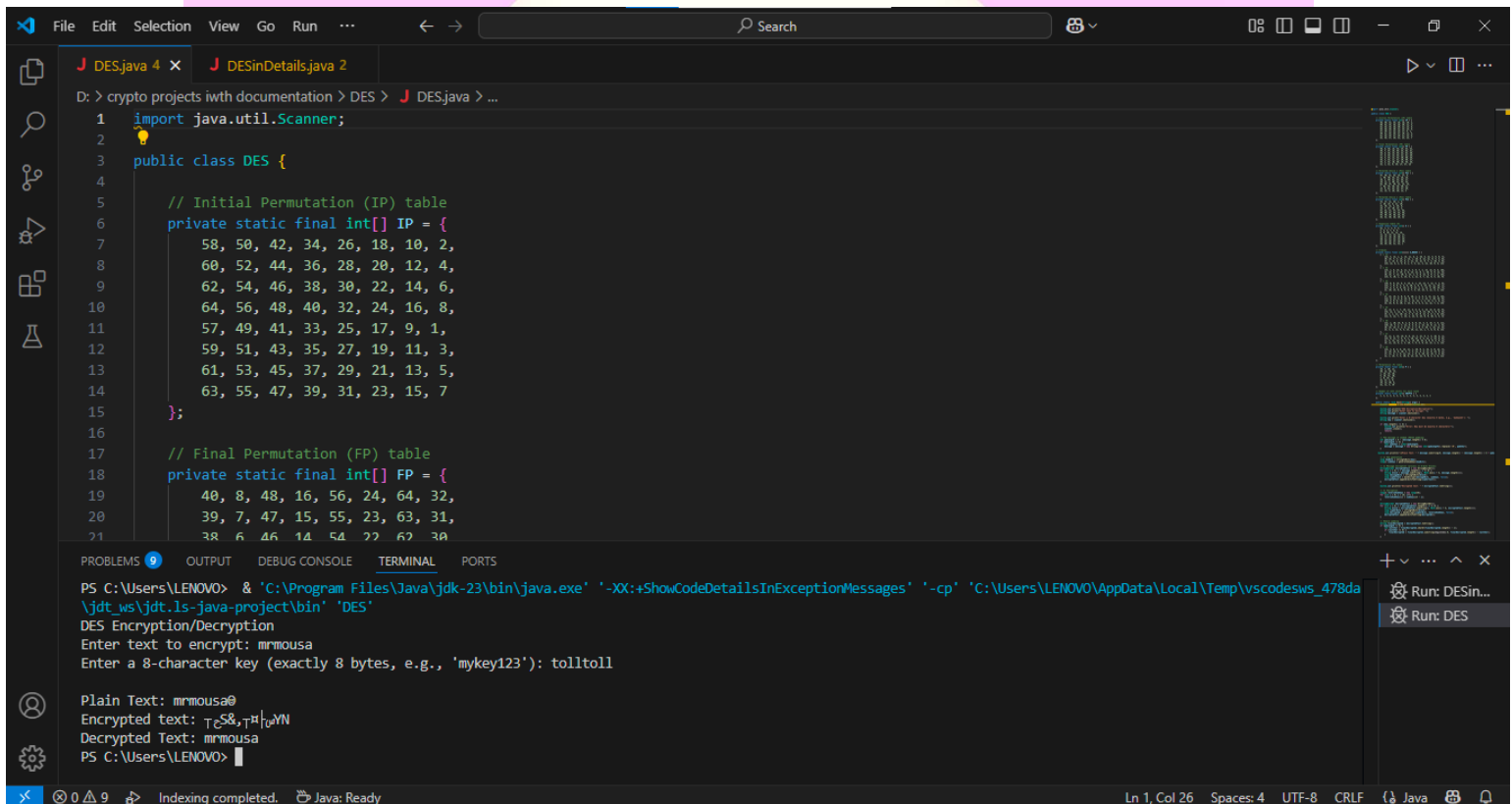
<https://linkedin.com/in/mousa123>

4. Decryption Process:

4.1 Same algorithm as encryption

4.2 Subkeys used in reverse order (K_{16} to K_1)

4.3 Final output is original plaintext



```
File Edit Selection View Go Run ... Search
J DES.java 4 x J DESinDetails.java 2
D: > crypto projects iwth documentation > DES > J DES.java > ...
1 import java.util.Scanner;
2
3 public class DES {
4
5     // Initial Permutation (IP) table
6     private static final int[] IP = {
7         58, 50, 42, 34, 26, 18, 10, 2,
8         60, 52, 44, 36, 28, 20, 12, 4,
9         62, 54, 46, 38, 30, 22, 14, 6,
10        64, 56, 48, 40, 32, 24, 16, 8,
11        57, 49, 41, 33, 25, 17, 9, 1,
12        59, 51, 43, 35, 27, 19, 11, 3,
13        61, 53, 45, 37, 29, 21, 13, 5,
14        63, 55, 47, 39, 31, 23, 15, 7
15    };
16
17    // Final Permutation (FP) table
18    private static final int[] FP = {
19        40, 8, 48, 16, 56, 24, 64, 32,
20        39, 7, 47, 15, 55, 23, 63, 31,
21        38, 6, 46, 14, 54, 22, 62, 30
22    };
23
24    public static void main(String[] args) {
25        Scanner scanner = new Scanner(System.in);
26        System.out.println("DES Encryption/Decryption");
27        System.out.print("Enter text to encrypt: ");
28        String text = scanner.nextLine();
29        System.out.print("Enter a 8-character key (exactly 8 bytes, e.g., 'mykey123'): ");
30        String key = scanner.nextLine();
31
32        // Encryption
33        String encryptedText = encrypt(text, key);
34        System.out.println("Encrypted text: " + encryptedText);
35
36        // Decryption
37        String decryptedText = decrypt(encryptedText, key);
38        System.out.println("Decrypted text: " + decryptedText);
39    }
40
41    // Encryption method
42    private static String encrypt(String text, String key) {
43        // ... (Encryption logic) ...
44    }
45
46    // Decryption method
47    private static String decrypt(String text, String key) {
48        // ... (Decryption logic) ...
49    }
50
51    // ... (Other methods) ...
52}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\LENOVO> & 'C:\Program Files\Java\jdk-23\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\LENOVO\AppData\Local\Temp\vscode\vs_478da...'
DES Encryption/Decryption
Enter text to encrypt: mmmousa
Enter a 8-character key (exactly 8 bytes, e.g., 'mykey123'): tolltoll

Plain Text: mmmousa@
Encrypted text: TcS&Tt|wYN
Decrypted Text: mmmousa
PS C:\Users\LENOVO>
```

Ln 1, Col 26 Spaces: 4 UTF-8 CRLF Java

Always remember:

1)Key generation:

- 1- permutation using pc1 table.
 - 2- divide each key into 2 each of 28 bits.
 - 3- now we have 16 subkeys "56 bits"
 - 4- therefore apply pc2 on concatenated subkeys to have 48 bits to work in fistel network.
- (These enter the function with R0 then XORed with L0 to get R1)

2)Message encryption:

- 1-change the message inputed by the user into binary
- 2- apply ip permutation
- 3-divide into 2 halves
- 4- $L_n = R_{n-1}$, $R_n = L_{n-1} + f(R_{n-1} , K_n)$, where n is the number of iteration
- 5- use expansion permutation no have 8 groups of 6 bits to have 48 bits with the 48 bits of the keys
- 6-Apply s boxes (now we have 8 groups of 4 bits)
- 7-Apply P permuation which gives us the ifnal value of f
- 8- xor with the L_{n-1}

<https://linkedin.com/in/mousa123>

Reference:

<https://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm>

