



Ministry of Education and Research of the Republic of
Moldova
Technical University of Moldova
Department of Software and Automation Engineering

REPORT

Laboratory work No. 4

Discipline: Cryptography and Security

Elaborated:

Gavriliuc Tudor, FAF - 221

Checked:

asist. univ., Nirca Dumitru

Chişinău 2024

Topic: Cypher Blocks. DES Algorithm

Tasks:

To develop a program in one of the preferred programming languages for implementing an element of the DES algorithm. The task will be chosen based on the student's ordinal number n in the group list, according to the formula: $nr_task = n \bmod 11$. For each task, the tables used and all intermediate steps must be displayed on the screen. The input data should either be user-provided or generated randomly.

2.2. Given K_i in the DES algorithm, determine C_i and D_i for a given i

Theoretical notes:

1. Initial Key Permutation (PC-1)

- **Purpose:** The initial permutation (PC-1) rearranges the bits of the 64-bit key K to reduce dependencies between input bits and their corresponding output bits.
- **Mechanism:**
 - A predefined table (PC-1) selects and permutes 56 bits from the 64-bit key.
 - The 8 parity bits (one for each byte) are ignored during this process.

Formula:

Permuted Key = PC-1(K)

Output:

- A 56-bit permuted key is divided into two halves:
 - C : The leftmost 28 bits.
 - D : The rightmost 28 bits.

2. Splitting into C and D Components

- The permuted 56-bit key is divided into two halves for independent manipulation:
 - C : Represents the left half of the key.
 - D : Represents the right half of the key.
- This separation allows for bitwise operations to be performed independently on each half.

3. Left Circular Shifts

- **Purpose:** To introduce variability and diffusion in the subkey generation.
- **Mechanism:**
 - CCC and DDD are each shifted left by a number of bits specified by the **shift schedule** for the current round i .
 - The shifts are **circular**, meaning bits that overflow the leftmost end are reintroduced at the rightmost end.

Formula:

$C_i = \text{LeftShift}(C_{i-1}, \text{shifts}[i])$
 $D_i = \text{LeftShift}(D_{i-1}, \text{shifts}[i])$

Shift Schedule:

- A predefined array dictates the number of shifts for each round i . For example:
 - Round 1: 1 shift.
 - Round 2: 1 shift.
 - Round 3: 2 shifts, and so on.

4. Intermediate Values

- The C_i and D_i values at each round are crucial for deriving the round-specific subkeys K_i .
- These intermediate values illustrate how the initial key evolves across rounds, introducing complexity and security into the DES encryption process.

5. Relevance in DES

The key scheduling process, including the steps to compute C_i and D_i , ensures:

- **Diffusion:** Small changes in the initial key propagate to all subkeys.
- **Unpredictability:** The use of a shift schedule and permutations increases the complexity of reverse-engineering subkeys.

Implementation:

1. generate_random_key()

Purpose: Generates a random 64-bit key, simulating a possible input for the DES encryption process.

Usage:

- Used when the user opts for automatic generation of a key.
- Ensures the key has random bit values (0 or 1).

Output: A 64-character binary string representing a 64-bit key.

2. apply_permutation(table, key)

Purpose: Applies a specific permutation to the input key based on a given table.

Usage:

- Implements the **PC-1 permutation** in the DES algorithm.
- Rearranges or selects bits from the input key based on indices provided in the table.

Parameters:

- table: A list of integers specifying the new positions of bits.
- key: The original key to be permuted (binary string).

3. left_shift(bits, shifts)

Purpose: Performs a circular left shift (rotation) on a binary string.

Usage:

- Shifts the bits in the CCC and DDD halves during key scheduling.
- Ensures bits that overflow on the left are reintroduced at the right.

Parameters:

- bits: Binary string representing the bits to be shifted.
- shifts: Number of positions to shift.

Here is a summary of the **key functions** implemented in the DES key scheduling task and their respective meanings:

1. generate_random_key()

Purpose: Generates a random 64-bit key, simulating a possible input for the DES encryption process.

Usage:

- Used when the user opts for automatic generation of a key.
- Ensures the key has random bit values (0 or 1).

Output: A 64-character binary string representing a 64-bit key.

2. apply_permutation(table, key)

Purpose: Applies a specific permutation to the input key based on a given table.

Usage:

- Implements the **PC-1 permutation** in the DES algorithm.
- Rearranges or selects bits from the input key based on indices provided in the table.

Parameters:

- table: A list of integers specifying the new positions of bits.
- key: The original key to be permuted (binary string).

Output: A permuted binary string.

3. left_shift(bits, shifts)

Purpose: Performs a circular left shift (rotation) on a binary string.

Usage:

- Shifts the bits in the CCC and DDD halves during key scheduling.
- Ensures bits that overflow on the left are reintroduced at the right.

Parameters:

- bits: Binary string representing the bits to be shifted.
- shifts: Number of positions to shift.

Output: A binary string that has been circularly shifted.

Example:

python

4. des_key_schedule(key_plus, i)

Purpose: Calculates C_i and D_i for a specific round i in the DES key schedule.

Usage:

- Implements the core of the key scheduling process.
- Applies the **PC-1 permutation** and performs **left shifts** for each round up to i .

Parameters:

- `key_plus`: The initial 64-bit key (binary string).
- `i`: The round number for which C_i and D_i are required.

Steps:

1. Apply **PC-1** to permute the key and split it into C_0 and D_0 .
2. Perform **circular left shifts** for each round up to i .
3. Display intermediate C and D values.

Output: The C_i and D_i values as binary strings.

Results:

```
Algoritmul DES: Calcularea C_i și D_i pentru un i dat.
Introduceți cheia manual (1) sau generați aleatoriu (2): 2
Cheia generată aleatoriu este: 10000011100100100100010111100100100101010101101101110000110110
Introduceți numărul rundeii (1-16): 1
Cheia după permutarea PC-1: 0101101101101100100010001111101000111111100010000000010
Cheia C0: 0101101101101100100010001111
Cheia D0: 1010001111111100010000000010
Runda 1: C1 = 1011011011011001000100011110, D1 = 0100011111111000100000000101
Rezultatul final pentru runda 1: C1 = 1011011011011001000100011110, D1 = 0100011111111000100000000101

Process finished with exit code 0
```

Conclusion

This implementation provides a detailed look into the DES key scheduling process. It includes step-by-step calculations and displays intermediate results for better understanding. The program allows for flexibility in input by either accepting a user-provided key or generating one randomly. This makes the implementation versatile and suitable for educational purposes.