Design and Implementation of a DES Encryption/Decryption System with Intermediate Round Verification

CSE350/550: Network Security (Jan–May 2025)
Programming Assignment No. 2
Due Date: [Insert Due Date]

Student Details:

- Pranjal Bharti, 2021080
- Daksh Bhasin, 201035

1. Introduction

The Data Encryption Standard (DES) is a symmetric-key algorithm that encrypts and decrypts data in fixed-size 64-bit blocks using a 64-bit key (with 56 bits effectively used for encryption). This project involves designing and implementing DES from scratch without relying on external cryptographic libraries. In addition to the core encryption and decryption functionality, the system captures intermediate states during the 16-round Feistel structure and verifies specific relationships between rounds. This report describes the design, implementation details, verification tests, and challenges encountered during the project.

2. System Overview and Design Objectives

The main objectives of the system are as follows:

- **Encryption/Decryption:** Implement the complete DES algorithm including key scheduling, initial and final permutations, expansion, S-box substitution, and permutation functions.
- **Intermediate State Capture:** Record the intermediate states after each round during both encryption and decryption.
- **Verification Requirements:**
    - **(a)** Verify that decryption of a ciphertext yields the original plaintext.

    - **(b)** Verify that the output of the 1st encryption round equals the output of the 15th decryption round (after accounting for the Feistel structure's swap of left and right halves).
    - **(c)** Verify that the output of the 14th encryption round equals the output of the 2nd decryption round (again, after swapping halves).

The system is implemented in Python, and it uses hexadecimal notation to represent 64-bit blocks of data, which is both standard practice in cryptography and convenient for displaying binary information.

3. Detailed System Design

3.1 DES Algorithm Overview

DES is a block cipher that operates on a 64-bit block of data using a series of operations:

- **Initial Permutation (IP):** Rearranges the bits of the plaintext.
- **16 Rounds of Processing:** Each round employs a Feistel structure where the block is split into two 32-bit halves. The right half is expanded, combined with a round key, processed through S-boxes, and permuted before being XORed with the left half. The halves are then swapped.
- **Final Permutation (IP^{-1}):** After 16 rounds, the halves are recombined (with a final swap) and permuted to yield the ciphertext.

3.2 Key Generation

The key generation process involves:

- **Permuted Choice 1 (PC-1):** Converting the 64-bit key to 56 bits by dropping parity bits.
- **Left Shifts:** Splitting the 56-bit key into two 28-bit halves and applying a left shift schedule over 16 rounds.
- **Permuted Choice 2 (PC-2):** Reducing the shifted 56-bit key to 48 bits to produce a round key for each round.

3.3 Encryption and Decryption Processes

The system implements both encryption and decryption:

- **Encryption:**
    1. Apply the initial permutation to the plaintext.
    2. Execute 16 rounds of the Feistel function where, in each round, the right half is expanded, XORed with the round key, substituted via S-boxes, and permuted before being combined with the left half.
    3. Swap the halves and apply the final permutation to generate the ciphertext.

- **Decryption:**
  The decryption process mirrors encryption, except that the round keys are used in reverse order. The same Feistel structure ensures that the algorithm is reversible.

3.4 Intermediate Round State Capture

To facilitate verification of specific properties:

- **Encryption Round States:** The system captures and stores the 64-bit state after each of the 16 encryption rounds.
- **Decryption Round States:** Similarly, the system records the state after each decryption round.

Due to the Feistel structure, the intermediate decryption states have their left and right halves in reversed order relative to the encryption states. Before comparing a decryption state to an encryption state, the system swaps the two halves.

4. Verification and Testing

The implementation is tested using at least three pairs of `<plaintext, ciphertext>` values. Each pair is defined using hexadecimal strings that represent 64-bit blocks. Some commonly used test vectors include:

- `"0123456789ABCDEF"`
- `"1234567890ABCDEF"`
- `"0000000000000000"`

4.1 Verification (a): Plaintext Integrity

For each test pair, the system encrypts the plaintext and then decrypts the resulting ciphertext. The decrypted output is compared with the original plaintext to ensure that the encryption-decryption process is lossless.

4.2 Verification (b): 1st Encryption Round vs. 15th Decryption Round

The system captures:

- **Round 1 of Encryption:** The state immediately after the first round.
- **Round 15 of Decryption:** The state captured after the 15th decryption round.

Before comparing, the decryption round state is adjusted by swapping its left and right halves to account for the mirror effect of the Feistel structure. The verification is successful if both states match.

4.3 Verification (c): 14th Encryption Round vs. 2nd Decryption Round

Similarly, the state after the 14th encryption round is compared with the state from the 2nd decryption round (after swapping halves). A match confirms that the DES algorithm's structure is correctly implemented and that the decryption process properly reverses the encryption process.

4.4 Screenshots of inputs and outputs

```
Test Pair 1:
Key:       133457799BBCDFF1
Plaintext: 0123456789ABCDEF
==============================
Ciphertext: 85E813540F0AB405
Decrypted:  0123456789ABCDEF
Verification (a): SUCCESS - Decrypted text matches the original plaintext.

1st Encryption Round:          F0AAF0AAEF4A6544
15th Decryption Round (swapped): F0AAF0AAEF4A6544
Verification (b): SUCCESS - 1st encryption round equals 15th decryption round (after swapping).

14th Encryption Round:         18C3155AC28C960D
2nd Decryption Round (swapped):  18C3155AC28C960D
Verification (c): SUCCESS - 14th encryption round equals 2nd decryption round (after swapping).

==============================
Test Pair 2:
Key:       AABB09182736CCDD
Plaintext: 1234567890ABCDEF
==============================
Ciphertext: 2482286C96BBD75F
Decrypted:  1234567890ABCDEF
Verification (a): SUCCESS - Decrypted text matches the original plaintext.

1st Encryption Round:          F0AAE8A5116BA133
15th Decryption Round (swapped): F0AAE8A5116BA133
Verification (b): SUCCESS - 1st encryption round equals 15th decryption round (after swapping).

14th Encryption Round:         E884876803AADF6C
2nd Decryption Round (swapped):  E884876803AADF6C
Verification (c): SUCCESS - 14th encryption round equals 2nd decryption round (after swapping).

==============================
Test Pair 3:
Key:       FFFFFFFFFFFFFFFF
Plaintext: 0000000000000000
==============================
Ciphertext: CAAAAF4DEAF1DBAE
Decrypted:  0000000000000000
Verification (a): SUCCESS - Decrypted text matches the original plaintext.

1st Encryption Round:          0000000038DBF9CB
15th Decryption Round (swapped): 0000000038DBF9CB
Verification (b): SUCCESS - 1st encryption round equals 15th decryption round (after swapping).

14th Encryption Round:         044D9D35472AC861
2nd Decryption Round (swapped):  044D9D35472AC861
Verification (c): SUCCESS - 14th encryption round equals 2nd decryption round (after swapping).
```

5. Challenges and Considerations

Several challenges were encountered during the design and implementation:

- **Implementation from Scratch:**
  Programming every component of DES—including key scheduling, permutations, and the Feistel function—requires careful handling of bit-level operations. Ensuring that each permutation table (such as IP, E, P, and the S-boxes) is correctly applied was critical.

- **Intermediate State Handling:**
  Capturing and comparing intermediate states across encryption and decryption rounds highlighted the importance of understanding the Feistel network. The need to swap halves during verification underscored the subtleties of the DES design.

- **Test Vector Selection:**
  Using hexadecimal test vectors is common practice in cryptography. Although the plaintexts (e.g., `"0123456789ABCDEF"`) might not be “readable” text, they serve as valid 64-bit blocks and are widely used in literature for validating DES implementations.

6. Conclusion

This project demonstrates a complete, self-contained implementation of the DES algorithm. By designing each component—from key generation to the encryption and decryption rounds—and incorporating intermediate state capture, the system not only encrypts and decrypts data correctly but also verifies crucial properties inherent in the DES Feistel structure. The successful verification of:

- Decryption yielding the original plaintext,
- The equivalence of the 1st encryption round with the 15th decryption round (after swapping halves), and
- The equivalence of the 14th encryption round with the 2nd decryption round (after swapping halves),

confirms the accuracy of the implementation. Future work could extend this project by adding support for other modes of operation, performance optimizations, or even integrating additional cryptographic algorithms for comparison.

End of Report