Simplified Data Encryption Standard (S-DES) - Overview

Luiz Fernando Sperandio David
190112735
Turma 01

[a] *Universidade de Brasília, Brasília, Brazil*

Abstract

The Simplified Data Encryption Standard (S-DES) is a simplified version of Data Encryption Standard (DES) designed for educational purposes.This article provides an overview of S-DES, including its key gerenartion, encryption and decryption processes. We discuss the theoritical fundation of S-DES and its pratical implementation. An experimental setup and analysis of results are also presented to evaluate the effectiveness of S-DES.

Keywords: S-DES, Cryptography

1. Introduction

The Simplified Data Encryption Standard (S-DES) was developed to help students understand the principles of the DES algorithm without the complexities of the full DES. S-DES operates on smaller key and block sizes, making it easier to understand and implement. This work aims to implement S-DES detailing the key generation, encryption, decryption processes, and to analyse its performace through experimental evaluation.

2. Theoritical Documentation

In this section, we will discuss the fundamental ideas related to computer security about S-DES. We will look at each of process in the next sections.

3. Key Generation Process

3.1. Initial Key Permutation (P10)

Let K be the initial 10-bit key represented as:

```
K = (k1, k2, k3, k4, k5, k6, k7, k8, k9, k10)
```

The permutation P10 is defined as:

```
P10(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10)
= (k3, k5, k2, k7, k4, k10, k1, k9, k8, k6)
```

This permutation can be concisely represented by the mapping:

```
P10 = {3, 5, 2, 7, 4, 10, 1, 9, 8, 6}
```

3.2. Circular Left Shift Operations

After the initial permutation, the 10-bit sequence undergoes two circular left shift operations:

1. First Shift (LS-1):
   - Split the 10-bit sequence into two 5-bit halves:
     ```
     Left5  = (1 0 0 0 0)
     Right5 = (0 1 1 0 0)
     ```
   - Perform a 1-bit circular left shift on each half:
     ```
     LS-1(Left5)  = (0 0 0 0 1)
     LS-1(Right5) = (1 1 0 0 0)
     ```

3.3. First Subkey (K1) Generation

The P8 permutation is applied to the concatenated result of LS-1 to generate K1:

```
P8 = {6, 3, 7, 4, 8, 5, 10, 9}
```

3.4. Second Subkey (K2) Generation

1. Second Shift (LS-2):
   - Using the result from LS-1, perform a 2-bit circular left shift on each half:
     ```
     LS-2(0 0 0 0 1) = (0 0 1 0 0)
     LS-2(1 1 0 0 0) = (0 0 0 1 1)
     ```
2. Apply P8:
   - Apply the P8 permutation to generate K2:
     ```
     LS-1(Left5)  = (0 0 0 0 1)
     LS-1(Right5) = (1 1 0 0 0)
     ```

*Corresponding author
Email address: 190112735@aluno.unb.br (Luiz Fernando Sperandio David
190112735
Turma 01)

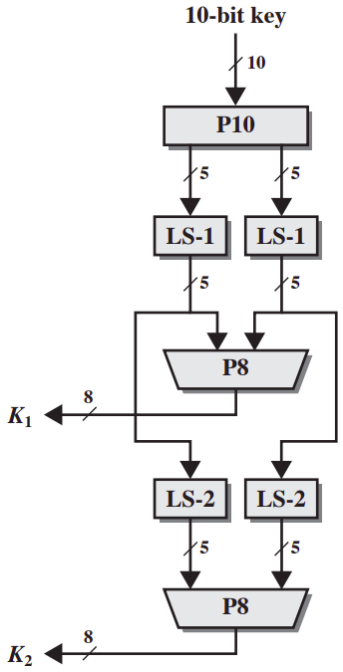

Figure 1: Key Generation for Simplified DES

3.5. Key Generation Summary

The complete key generation process can be summarized as:

1. Apply P10 to the initial 10-bit key
2. Perform LS-1 on both halves
3. Apply P8 to generate K1
4. Perform LS-2 on the results from step 2
5. Apply P8 to generate K2

This process ensures that each subkey is a unique permutation derived from the original key, providing the necessary security properties for the S-DES algorithm.

4. Encryption Process

The encryption process in S-DES transforms an 8-bit block of plaintext into an 8-bit block of ciphertext using the generated subkeys K1 and K2. This section provides a detailed explanation of each step involved in the encryption process.

4.1. Initial Permutation (IP)

The plaintext undergoes an initial permutation (IP) to rearrange the bits before the encryption rounds begin. The IP is defined as:

$$IP = \{2, 6, 3, 1, 4, 8, 5, 7\}$$

For example, if the plaintext is `11010111`:

```
Input:  1 1 0 1 0 1 1 1
IP:     1 1 0 1 1 1 0 1
Output: 11011101
```

4.2. Splitting

After the initial permutation, the 8-bit block is split into two 4-bit halves:

```
Left (L_0):  1101
Right (R_0): 1101
```

4.3. First Round of Encryption

The first round of encryption involves several sub-steps:

1. Expansion/Permutation (E/P)
   - The right half (R0) is expanded and permuted to 8 bits using E/P:

     $$E/P = \{4, 1, 2, 3, 2, 3, 4, 1\}$$

   - For R0 = `1101`, the result is:

     ```
     E/P(R0) = 11101011
     ```

4.4. Key Mixing

The expanded right half is XORed with the first subkey K1:

```
K1 = 10100100
XOR: 11101011 +(shore) 10100100 = 01001111
```

4.5. Substitution (S-Boxes)

The 8-bit result is divided into two 4-bit halves and passed through two S-Boxes (S1 and S2). Each S-Box maps a 4-bit input to a 2-bit output.

The most complex component of S-DES is the function fK, which consists of a combination of permutation and substitution functions. The functions can be expressed as follows. Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to fK, and let F be a mapping (not necessarily one to one) from 4-bit strings to 4-bit strings. Then we let fK(L, R) = (L (+) F(R, SK), R), where SK is a subkey and (+) is the bit-by-bit exclusive-OR function.

For example, suppose the output of the IP stage is (10111101) and F(1101, SK) = (1110) for some key SK. Then fK(10111101) = (01011101) because (1011) (+) (1110) = (0101).

We now describe the mapping F. The input is a 4-bit number (n1n2n3n4). The first operation is an expansion/permutation operation:

$$E/P = \{4, 1, 2, 3, 2, 3, 4, 1\}$$

For what follows, it is clearer to depict the result in this fashion:

$$\begin{array}{cccc} n4 & |n1 & n2 & |n3 \\ n2 & |n3 & n4 & |n1 \end{array}$$

The 8-bit subkey K1 = (k11, k12, k13, k14, k15, k16, k17, k18) is added to this value using exclusive-OR:

$$\begin{array}{cccc} n4(+)k11| & n1(+)k12 & n2(+)k13| & n3(+)k14 \\ n2(+)k15| & n3(+)k16 & n4(+)k17| & n1(+)k18 \end{array}$$

Let us rename these 8 bits:

$$\begin{array}{cccc} p0,0| & p0,1 & p0,2| & p0,3 \\ p1,0| & p1,1 & p1,2| & p1,3 \end{array}$$

The first 4 bits (first row of the preceding matrix) are fed into the S-box S0 to produce a 2-bit output, and the remaining 4 bits (second row) are fed into S1 to produce another 2-bit output. These two boxes are defined as follows:

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

The S-boxes operate as follows. The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the S-box. The entry in that row and column, in base 2, is the 2-bit output. For example, if (p0,0p0,3) = (00) and (p0,1p0,2) = (10), then the output is from row 0, column 2 of S0, which is 3, or (11) in binary. Similarly, (p1,0p1,3) and (p1,1p1,2) are used to index into a row and column of S1 to produce an additional 2 bits.

Next, the 4 bits produced by S0 and S1 undergo a further permutation as follows:

$$P4 = \{2, 4, 3, 1\}$$

The output of P4 is the output of the function F.

- **S1**:

```
Input: 0100
Output: 01 (assuming S1 mapping)
```

- **S2**:

```
Input: 1111
Output: 10 (assuming S2 mapping)
```

4.6. Permutation (P4)

The combined 4-bit output from the S-Boxes is permuted using P4:

$$P4 = \{2, 4, 3, 1\}$$

For the combined output `0110`:

```
P4(0110) = 1010
```

4.7. XOR with Left Half (L0)

The permuted result is XORed with the left half (L0):

```
L0 = 1101
XOR: 1101 (+) 1010 = 0111
```

The result forms the new left half for the next round.

4.8. Swapping (SW)

The halves are swapped:

```
L1 = R0 = 1101
R1 = 0111
```

4.9. Second Round of Encryption

The second round repeats the steps of the first round but uses the second subkey K2. The process involves:

- Expansion/Permutation (E/P)
- Key Mixing with K2
- Substitution (S-Boxes)
- Permutation (P4)

- XOR with Left Half (L_1)

The new results form the final combined halves.

4.10. Final Permutation (IP^{-1})

The final combined 8-bit block undergoes the inverse permutation (IP^{-1}) to produce the ciphertext:

$$IP^{-1} = \{4, 1, 3, 5, 7, 2, 8, 6\}$$

For the final block `00100011`:

```
Input:  00100011
IP$^{-1}$:  10101000
```

5. Decryption Process

The decryption process in S-DES reverses the encryption process to transform an 8-bit block of ciphertext back into an 8-bit block of plaintext using the same subkeys K1 and K2, but in reverse order. This section simplifies the explanation by referring to the steps outlined in the encryption process (Section 4) and highlighting the key differences.

5.1. Initial Permutation (IP)

Similar to the encryption process, the ciphertext undergoes an initial permutation (IP) using the same permutation table:

$$IP = \{2, 6, 3, 1, 4, 8, 5, 7\}$$

For example, if the ciphertext is `10101000`:

```
Input:  1 0 1 0 1 0 0 0
IP:     0 0 1 1 0 0 1 0
Output: 00110010
```

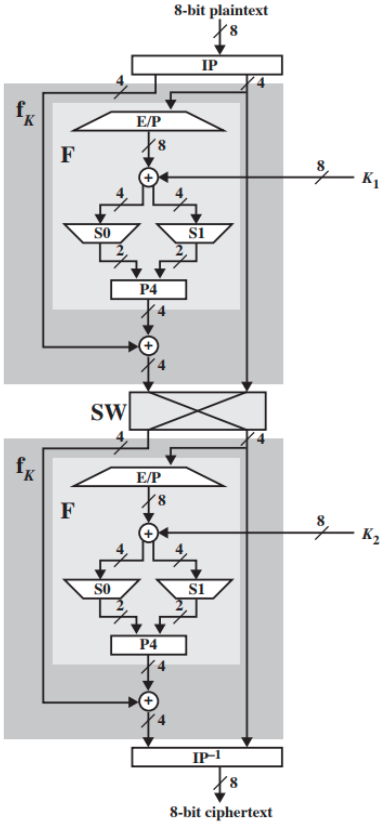

Figure 2: Simplified DES Encryption Detail

5.2. Splitting

After the initial permutation, the 8-bit block is split into two 4-bit halves:

```
Left ($L_0$):  0011
Right ($R_0$): 0010
```

5.3. First Round of Decryption

The first round of decryption is similar to the first round of encryption but uses the second subkey (K2) instead of the first subkey (K1). The steps include:

- **Expansion/Permutation (E/P)**
- **Key Mixing**

```
K2 = 01000011
XOR: 00011001 \oplus 01000011 = 01011010
```

- **Substitution (S-Boxes)**
- **Permutation (P4)**
- **XOR with Left Half (L_0)**

```
$L_0$ = 0011
XOR: 0011 \oplus 1010 = 1001
```

The result forms the new left half for the next round.

5.4. Swapping

The halves are swapped:

```
$L_1$ = $R_0$ = 0010
$R_1$ = 1001
```

5.5. Second Round of Decryption

The second round of decryption again follows similar steps to the encryption process but uses K1. The process includes:

- Expansion/Permutation (E/P)
- Key Mixing with K1
- Substitution (S-Boxes)
- Permutation (P4)
- XOR with Left Half (L_1)

The new results form the final combined halves.

5.6. Final Permutation (IP^{-1})

The final combined 8-bit block undergoes the inverse permutation (IP^{-1}) to produce the plaintext:

$$IP^{-1} = \{4, 1, 3, 5, 7, 2, 8, 6\}$$

For the final block `11011101`:

```
Input:  11011101
IP$^{-1}$:  11010111
```

This completes the decryption process, producing the plaintext.

6. Results Analysis

6.1. Key Generation Analysis

The implementation results demonstrate the practical application of the key generation process described in Section 3 Using an initial 10-bit key K = 1010000010, we can trace the complete key generation process.

6.1.1. Initial Permutation (P10) Validation

As described in Section 3.1, the P10 permutation was applied to the initial key:

```
Input key:      1010000010
After P10:      1000001100
```

This result confirms the correct implementation of the P10 permutation table 3,5,2,7,4,10,1,9,8,6, as the output matches our theoretical expectations.

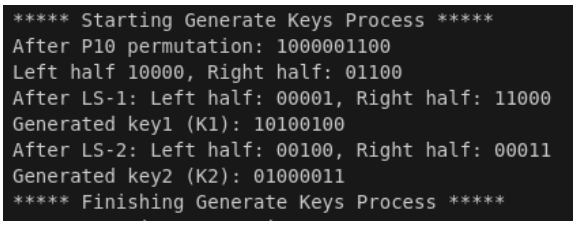

Figure 3: Key Generation Analysis output

6.1.2. Split and Shift Operations

Following the process outlined in Section 3.2, the implementation shows:

```
    Left half:      10000
    Right half:     01100
After LS-1:
    Left half:      00001
    Right half:     11000
```

The output demonstrates the accurate execution of the 1-bit circular left shift (LS-1) on both halves.

6.1.3. Subkey Generation

As detailed in Sections 3.3 and 3.4, the implementation generated two subkeys:

```
    K1 (First subkey):  10100100
    K2 (Second subkey): 01000011
```

These results validate the correct application of:

- P8 permutation after LS-1 for K1
- LS-2 operation:

```
    Left half:      00100
    Right half:     00011
```

- P8 permutation for K2 generation

6.2. Encryption Process Validation

The implementation demonstrates a complete encryption cycle using:

```
    Plaintext:          11010111
    Key:                1010000010
```

The encryption process follows these steps with corresponding outputs:

1. Initial Permutation (IP):

```
    Input:          11010111
    After IP:       11011101
```

2. Split Operation:

```
    Left half:      1101
    Right half:     1101
```

3. Round Operations:

```
    After first round (K1):
    Left:           1101
    Right:          0010

    After second round (K2):
    Left:           0010
    Right:          0011
```

4. Final Permutation:

```
    Before IP^-1:   00100011
    Ciphertext:     10101000
```

6.3. Verification of Reversibility

The implementation confirms the algorithm's reversibility by demonstrating:

```
    Original Plaintext:  11010111
    Ciphertext:          10101000
    Decrypted Result:    11010111
```

This validates that:

1. The decryption process successfully reverses the encryption
2. The subkey application order is correctly reversed (K2 then K1)
3. All permutations and transformations are properly inverted

6.4. Implementation Correctness

The results demonstrate several critical aspects of correct implementation:

1. Key generation process maintains bit integrity throughout all transformations
2. Permutation operations (P10, P8, IP, IP-^{1}) are correctly implemented
3. The circular shift operations (LS-1, LS-2) maintain the required bit positions
4. The encryption and decryption processes are perfect inverses of each other

These results validate both the theoretical framework presented in Section 4 and its practical implementation, confirming the correctness of the S-DES algorithm implementation.

```
***** Starting Encryption Process *****
Encrypting plaintext: 11010111
After initial permutation (IP): 11011101
Initial split: Left: 1101, Right: 1101
After first round with K1: Left: 1101, Right: 0010
After second round with K2: Left: 0010, Right: 0011
After final permutation (IP-1): 10101000
***** Ending Encryption Process *****
```

Figure 4: Encryption Process Validation Output

Bibliographic

1. Appendix G Simplified DES:
   `http://mercury.webster.edu/aleshunas/COSC205130/G-SDES.pdf`
2. GitHub S-DES Implementetion in JAVA:
   `github.com/akshaychavan7/Simplified-Data-Encryption-Standard-S-DES-Algorithm`

The link for this article project on GitHub:
S-DES Implementation