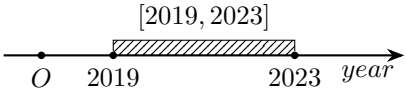
Beijing-Dublin International College

BDIC3024J Design Patterns

Software Engineering



Security & Privacy 2022
DES Algorithm Implementation

Student Name:
Yuyang Wang

Student ID:
19206226 & 19372316

October 12, 2022

Contents

1 Introduction

1.1 Data Encryption Standard (DES)

The following Data Encryption Algorithm (DES) and Triple Data Encryption Algorithm shall make up the Data Encryption Standard (DES) (TDEA, as described in ANSI X9.52). These devices must be built so that they may be utilized in a computer system or network to secure binary-coded data using cryptography. The application and environment will determine the implementation strategy. The devices must be developed in a way that allows for testing and validation that they accurately carry out the transformations outlined in the ensuing algorithms.

1.2 Implemented Functionalities

- Implemented **both** the encryption and the decryption function.
- **No restriction** on the length of the plain/cipher text or the length of the key.
- Designed and implemented a **user-friendly** interface.
- Plain/cipher text can be **drawn from the file system** or **input through the interface**.
- Encryption and decryption are two **independent** components.
- **Key** needs to be supplied when performing either component.

1.3 Data Encryption Algorithm

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation IP, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation IP^{-1} . The key-dependent computation can be simply defined in terms of a function f, called the cipher function, and a function KS, called the key schedule. A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the use of the algorithm for decipherment is described. Finally, a definition of the cipher function f is given in terms of primitive functions which are called the selection functions S_i and the permutation function P. S_i, P and KS of the algorithm are contained Appendix.

2 Algorithm

2.1 Encryption method

Initial Permutation:

First, subscript substitution is used to modify the input's 64 bits, and the transformation matrix IP is:

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

Divide the output 64 bits into two 32-bit matrices by high 32 bits and low 32 bits, i.e.

$$(L_0, R_0) = IP(I) \tag{2}$$

Perform 16 iterations with the following rules:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f\left(R_{i-1}, K_{i-1}\right) \end{aligned} \tag{3}$$

Where the K_n is defined as:

$$K_n = KS(n, KEY) \tag{4}$$

The data after 16 iteration like this is subjected to another subscript substitution transformation, and its transformation matrix IP^{-1} is

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

2.2 Key Schedule

The key generation input of the DES algorithm is a 64-bit master key with 8 bits as parity bits and the output is 16 48-bit subkeys.

Permuted choice 1 ($PC1$) is determined by the following table:

$$PC1 = \begin{pmatrix} 57 & 49 & 41 & 33 & 25 & 17 & 9 \\ 1 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 2 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 3 & 60 & 52 & 44 & 36 \\ & & & & & & \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 7 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 6 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 5 & 28 & 20 & 12 & 4 \end{pmatrix} \tag{6}$$

This first part of this matrix determines how the bits of C_0 and the second part defines how D_0 are defined.

In terms of C_n and D_n, they are obtained from C_{n-1} and D_{n-1}. For $n = 1, 2, 3, \ldots, 16.$, they are accomplished by the following schedule of left shifts of the individual blocks. The LS matrix is defined as:

$$L = \begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \end{pmatrix} \tag{7}$$

After this, the key will go through another permuted choice, here the Permuted choice 2 is defined as:

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

By doing this we can obtain 16 different keys.

2.3 F Function

First, a selection expansion operation is performed to transform the input 32 bits by subscript substitution to obtain a 48-bit matrix whose transformation matrix E is:

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

By doing this, the result of $E(R)$ will be a 48 bit matrix.

Then, the obtained 48-bit data and the corresponding subkeys are subjected to a bit-by-bit addition modulo 2 operation, the result $B = K \oplus E(R)$.

After this, the 48 bits obtained after the operation are divided into eight 6-bit matrices for the selected compression transformation - i.e., the S-box transformation, whose eight S-boxes are shown below

$$S_1 = \begin{pmatrix} 14 & 4 & 13 & 1 & 2 & 15 & 11 & 8 & 3 & 10 & 6 & 12 & 5 & 9 & 0 & 7 \\ 0 & 15 & 7 & 4 & 14 & 2 & 13 & 1 & 10 & 6 & 12 & 11 & 9 & 5 & 3 & 8 \\ 4 & 1 & 14 & 8 & 13 & 6 & 2 & 11 & 15 & 12 & 9 & 7 & 3 & 10 & 5 & 0 \\ 15 & 12 & 8 & 2 & 4 & 9 & 1 & 7 & 5 & 11 & 3 & 14 & 10 & 0 & 6 & 13 \end{pmatrix} \tag{10}$$

$$S_2 = \begin{pmatrix} 15 & 1 & 8 & 14 & 6 & 11 & 3 & 4 & 9 & 7 & 2 & 13 & 12 & 0 & 5 & 10 \\ 3 & 13 & 4 & 7 & 15 & 2 & 8 & 14 & 12 & 0 & 1 & 10 & 6 & 9 & 11 & 5 \\ 0 & 14 & 7 & 11 & 10 & 4 & 13 & 1 & 5 & 8 & 12 & 6 & 9 & 3 & 2 & 15 \\ 13 & 8 & 10 & 1 & 3 & 15 & 4 & 2 & 11 & 6 & 7 & 12 & 0 & 5 & 14 & 9 \end{pmatrix} \tag{11}$$

$$S_3 = \begin{pmatrix} 10 & 0 & 9 & 14 & 6 & 3 & 15 & 5 & 1 & 13 & 12 & 7 & 11 & 4 & 2 & 8 \\ 13 & 7 & 0 & 9 & 3 & 4 & 6 & 10 & 2 & 8 & 5 & 14 & 12 & 11 & 15 & 1 \\ 13 & 6 & 4 & 9 & 8 & 15 & 3 & 0 & 11 & 1 & 2 & 12 & 5 & 10 & 14 & 7 \\ 1 & 10 & 13 & 0 & 6 & 9 & 8 & 7 & 4 & 15 & 14 & 3 & 11 & 5 & 2 & 12 \end{pmatrix} \tag{12}$$

$$S_4 = \begin{pmatrix} 7 & 13 & 14 & 3 & 0 & 6 & 9 & 10 & 1 & 2 & 8 & 5 & 11 & 12 & 4 & 15 \\ 13 & 8 & 11 & 5 & 6 & 15 & 0 & 3 & 4 & 7 & 2 & 12 & 1 & 10 & 14 & 9 \\ 10 & 6 & 9 & 0 & 12 & 11 & 7 & 13 & 15 & 1 & 3 & 14 & 5 & 2 & 8 & 4 \\ 3 & 15 & 0 & 6 & 10 & 1 & 13 & 8 & 9 & 4 & 5 & 11 & 12 & 7 & 2 & 14 \end{pmatrix} \tag{13}$$

$$S_5 = \begin{pmatrix} 2 & 12 & 4 & 1 & 7 & 10 & 11 & 6 & 8 & 5 & 3 & 15 & 13 & 0 & 14 & 9 \\ 14 & 11 & 2 & 12 & 4 & 7 & 13 & 1 & 5 & 0 & 15 & 10 & 3 & 9 & 8 & 6 \\ 4 & 2 & 1 & 11 & 10 & 13 & 7 & 8 & 15 & 9 & 12 & 5 & 6 & 3 & 0 & 14 \\ 11 & 8 & 12 & 7 & 1 & 14 & 2 & 13 & 6 & 15 & 0 & 9 & 10 & 4 & 5 & 3 \end{pmatrix} \tag{14}$$

$$S_6 = \begin{pmatrix} 12 & 1 & 10 & 15 & 9 & 2 & 6 & 8 & 0 & 13 & 3 & 4 & 14 & 7 & 5 & 11 \\ 10 & 15 & 4 & 2 & 7 & 12 & 9 & 5 & 6 & 1 & 13 & 14 & 0 & 11 & 3 & 8 \\ 9 & 14 & 15 & 5 & 2 & 8 & 12 & 3 & 7 & 0 & 4 & 10 & 1 & 13 & 11 & 6 \\ 4 & 3 & 2 & 12 & 9 & 5 & 15 & 10 & 11 & 14 & 1 & 7 & 6 & 0 & 8 & 13 \end{pmatrix} \tag{15}$$

$$S_7 = \begin{pmatrix} 4 & 11 & 2 & 14 & 15 & 0 & 8 & 13 & 3 & 12 & 9 & 7 & 5 & 10 & 6 & 1 \\ 13 & 0 & 11 & 7 & 4 & 9 & 1 & 10 & 14 & 3 & 5 & 12 & 2 & 15 & 8 & 6 \\ 1 & 4 & 11 & 13 & 12 & 3 & 7 & 14 & 10 & 15 & 6 & 8 & 0 & 5 & 9 & 2 \\ 6 & 11 & 13 & 8 & 1 & 4 & 10 & 7 & 9 & 5 & 0 & 15 & 14 & 2 & 3 & 12 \end{pmatrix} \tag{16}$$

$$S_8 = \begin{pmatrix} 13 & 2 & 8 & 4 & 6 & 15 & 11 & 1 & 10 & 9 & 3 & 14 & 5 & 0 & 12 & 7 \\ 1 & 15 & 13 & 8 & 10 & 3 & 7 & 4 & 12 & 5 & 6 & 11 & 0 & 14 & 9 & 2 \\ 7 & 11 & 4 & 1 & 9 & 12 & 14 & 2 & 0 & 6 & 10 & 13 & 15 & 3 & 5 & 8 \\ 2 & 1 & 14 & 7 & 4 & 10 & 8 & 13 & 15 & 12 & 9 & 0 & 3 & 5 & 6 & 11 \end{pmatrix} \tag{17}$$

After completing the operation of the S-box, the output of the S-box is then reassembled into a 32-bit matrix with one coordinate permutation, and the permutation P is:

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

3 Implementation

3.1 Technological Stack

- GUI: HTML + CSS + Bootstrap
- Logic: Javascript
- Environment: Chrome, Firefox

3.2 Implementation of Key Features

3.2.1 DES Algorithm

The implementation of DES algorithm are stored in the DES.JS file.

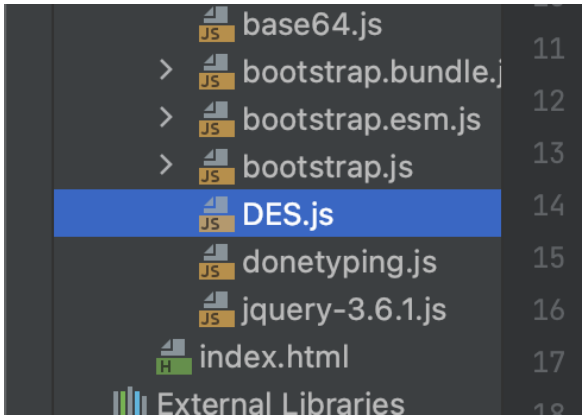


Figure 1: DES.js

3.2.2 Key Generation

```
function keySchedule(key) {
    // Generate the 16 keys for the 16 rounds
    // subkeys saves the 16 keys
    let subkeys = [];
    let perm = PC1.map(index => key[index - 1]).join("");
    // Split the permuted key into two halves
    // Left half
    let C0 = perm.substr(0, perm.length / 2);
    // Right half
    let D0 = perm.substr(perm.length / 2);
    let prevC0 = C0, prevD0 = D0;
    // Generate the 16 keys
    NUM_OF_LEFT_SHIFTS.forEach((shift, i) => {
        // Shift the left half
        C0 = shiftString(prevC0, shift);
        // Shift the right half
        D0 = shiftString(prevD0, shift);
        prevC0 = C0;
```

```
        prevD0 = D0;
        let pair = C0 + D0;
        // Generate the 16 keys
        subkeys.push(PC2.map(index => pair[index - 1]).join(""));
    });

    return subkeys;
}
```

3.2.3 XOR Operation

```
const stringXOR = (str1, str2, len) => {
    // XOR two strings
    let xor = Array(len);
    // Loop through the strings
    for (let i = 0; i < len; i++) {
        // XOR the two strings
        xor[i] = (str1[i] === str2[i] ? 0 : 1);
    }
    return xor.join("");
}
```

3.2.4 S-Box Operation

```
function sBoxOutput(bits) {
    // Get the output of the S-Boxes and join them
    return chunkString(bits, 6).map(function (group, sBox) {
        // Get the row and column of the S-Box
        let row = parseInt(group[0] + group[5], 2);
        let col = parseInt(group.slice(1, 5), 2);
        // Get the output of the S-Box
        return decToBin(S[sBox][16 * row + col]);
    }).join("");
}
```

3.2.5 DES Algorithm

```
function DES(msg, key, subkeys) {
    // Encrypt the message
    let perm = IP.map(index => msg[index - 1]).join(""); // init permute (
        IP)
    // Split the permuted message into two halves
    let L0 = perm.substr(0, perm.length / 2); // Left half
    let R0 = perm.substr(perm.length / 2); // Right half
    let prevL0 = L0, prevR0 = R0;
    for (let i = 0; i < 16; i++) {
        L0 = prevR0;
        // Get the output of the S-Boxes
        let sBoxOut = sBoxOutput(stringXOR(subkeys[i], expandBlock(R0),
            48));
        // Perform Permutation using the P-Table
        let finalPerm = P.map(index => sBoxOut[index - 1]).join("");
        // XOR the left half with the output of the S-Boxes
        R0 = stringXOR(prevL0, finalPerm, 32);
        prevL0 = L0;
        prevR0 = R0;
    }
    // Join the two halves
    let pair = R0 + L0;
```

```
    // Perform the final permutation
    let enc = FINAL_IP.map(index => pair[index - 1]).join("");
    // Return the encrypted message
    return chunkString(enc, 4).map(binToHex).join("").toUpperCase();
}
```

3.2.6 Encryption and Decryption function

```
function DESEncrypt(msg, key) {
    // Encapsulate the encryption process
    return DES(msg, key, keySchedule(key));
}

function DESDecrypt(msg, key) {
    // Encapsulate the decryption process
    return DES(msg, key, keySchedule(key).reverse());
}
```

4 Instructions of Using this Program

Base on the above algorithm, I build my own implementation of DES algorithm. I also deployed this program to http://enc.echo.cool/. You can preview this project using this link. But please use the code submitted when assessing this project.

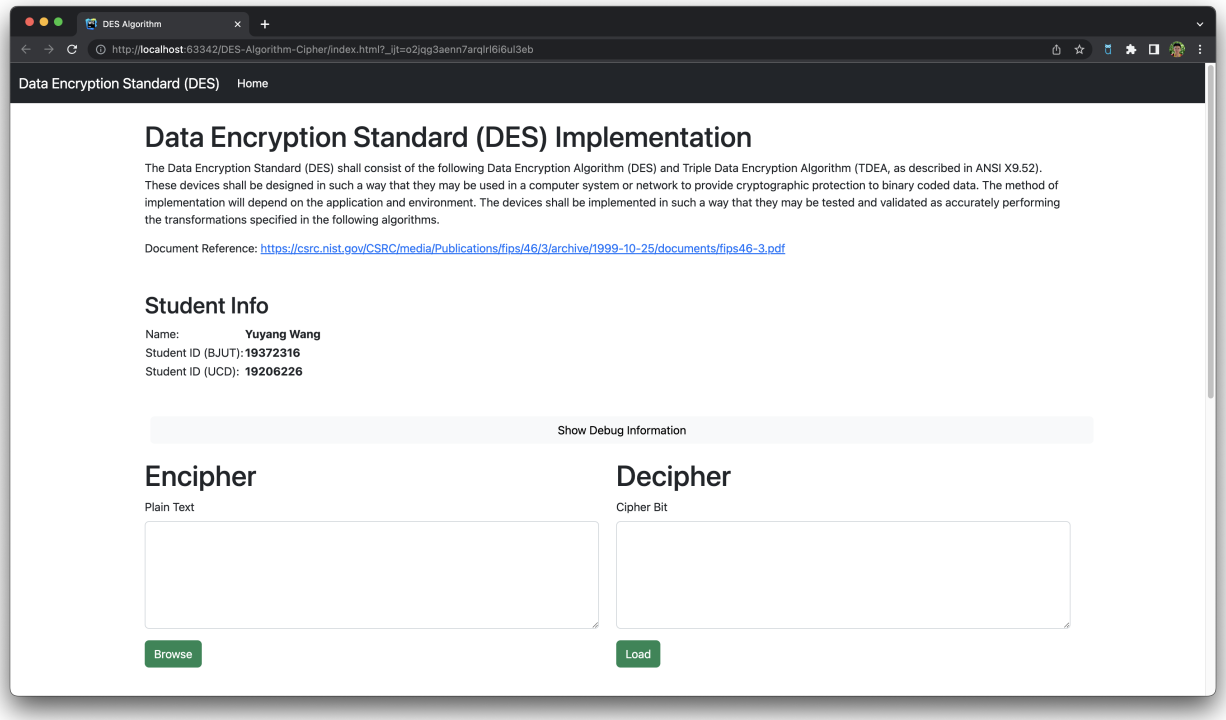


Figure 2: Overview of my Program

4.1 Enciphering

The left side of my program is the encipher part. To encipher a part of text, please input the content to the "Plain Text" input box. This program will automatically detect the change in this box and execute the encipher process.

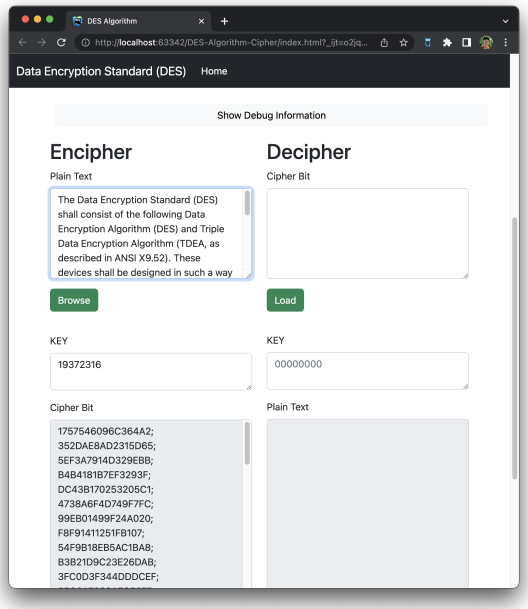


Figure 3: Enciphering

4.2 Deciphering

The right side of my program is the deciphering part, you can input ciphered text in the "Cipher Bit" area, this program will decipher this cipher text.

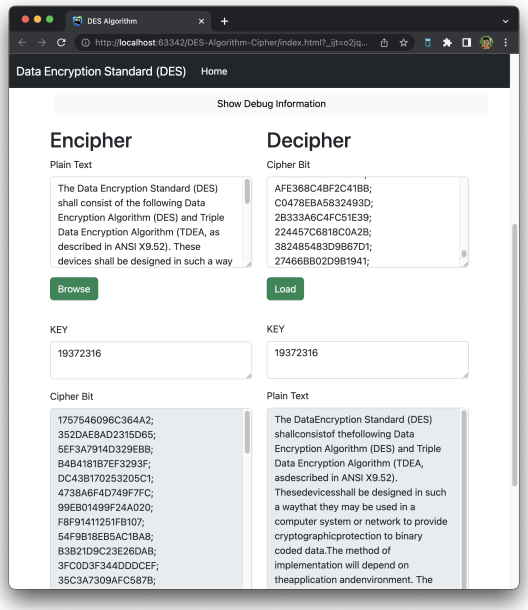


Figure 4: Deciphering (With Correct KEY)

You need to make sure that the password (key) is correct, otherwise the text will not be able correctly deciphered.

4.3 File Enciphering & Deciphering

Apart from plain text, you may upload file to this program to get a enciphered file.

You can click on the Browse button, and then you may choose a file.

After this, this program will automatically process the file and produce ciphered data. You can click the Download button to download enciphered file.

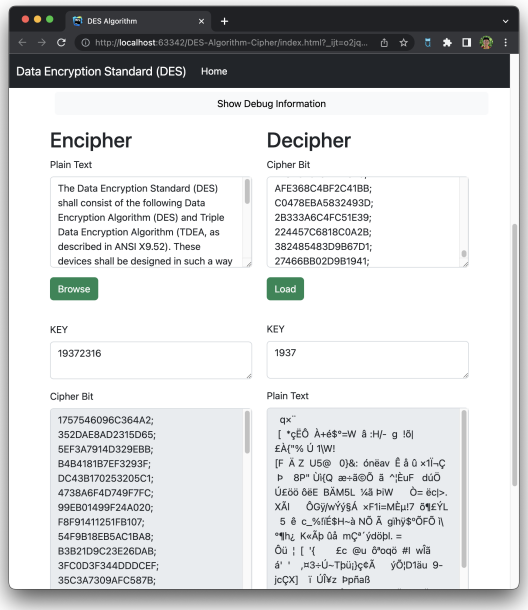


Figure 5: Deciphering (With Incorrect KEY)

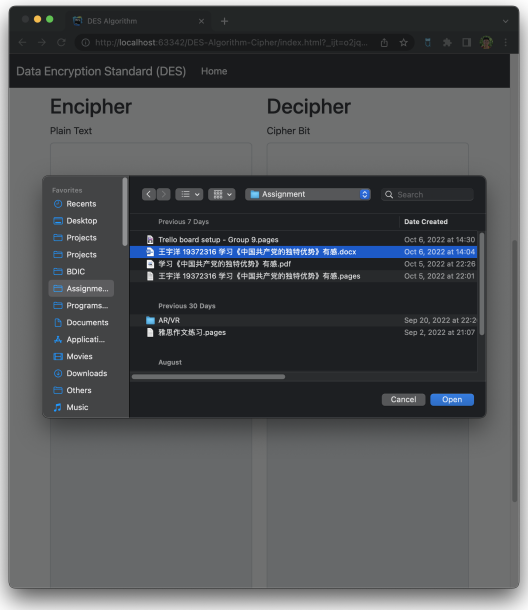


Figure 6: Choose File

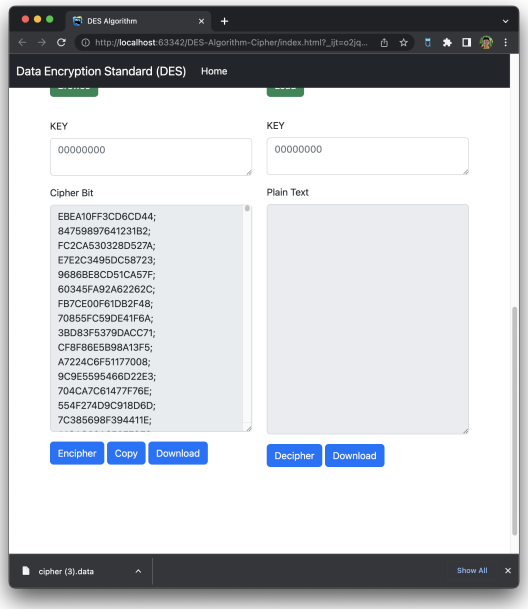


Figure 7: Download Ciphered Text

In terms of deciphering, you need to click on the load button. Then choose the cipher data file.

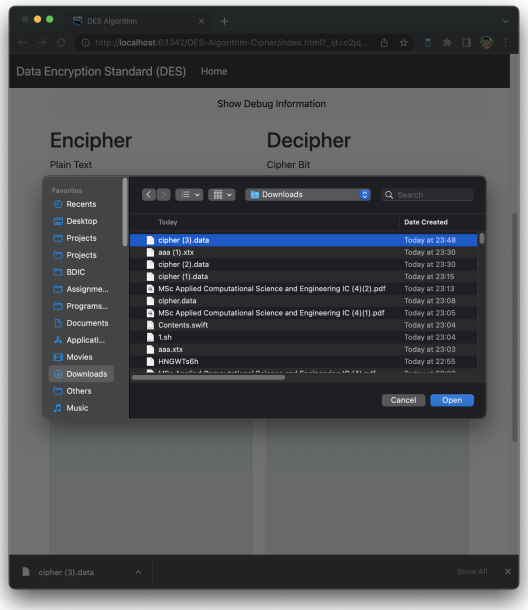


Figure 8: Choose Ciphered Data File

Then this program will automatically decipher the file, after the process is completed. You can click on the download button on the right side to download the decipered file.

4.4 Debug Function

This program provides Debug function, you can click into Show Debug Information to see the detailed operation process of the program.

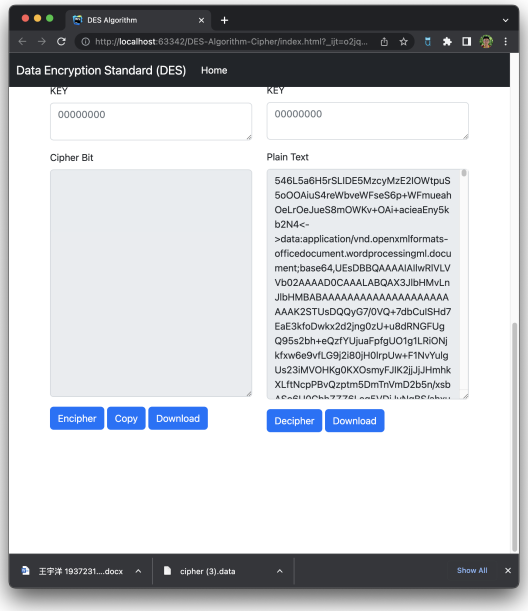


Figure 9: Download Deciphered File

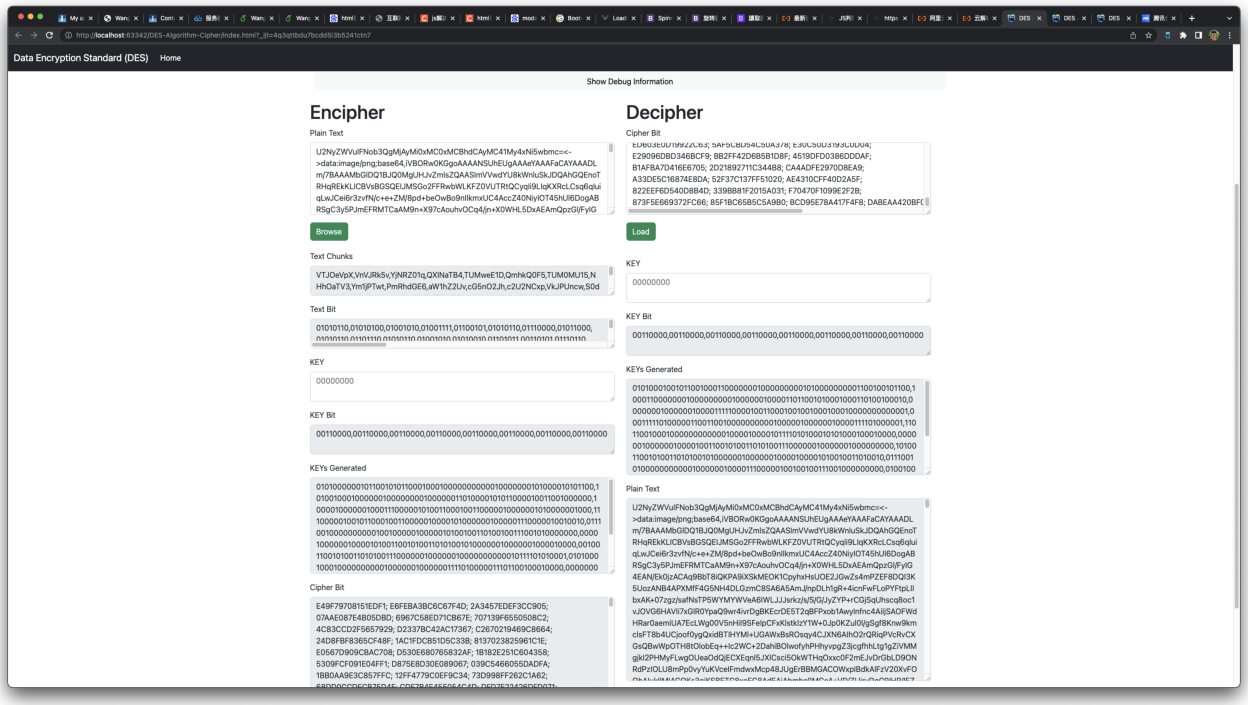


Figure 10: Debug Information