


## MỤC LỤC

BÀI 3: MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI .....	2
1.1. Thuật toán mã hóa DES .....	2
1.2. Thuật toán mã hóa AES .....	11
1.3. Bài tập thực hành .....	19

Trường ĐH CNTP TP.HCM  Khoa: Công nghệ thông tin  Bộ môn: Hệ thống thông tin  Môn: TH Mã hóa và ứng dụng	<p style="text-align: center;"><b>BÀI 3:</b> <b>MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI</b></p>	
--	--	---

## A. MỤC TIÊU

*Sau khi học xong bài này người học có khả năng:*

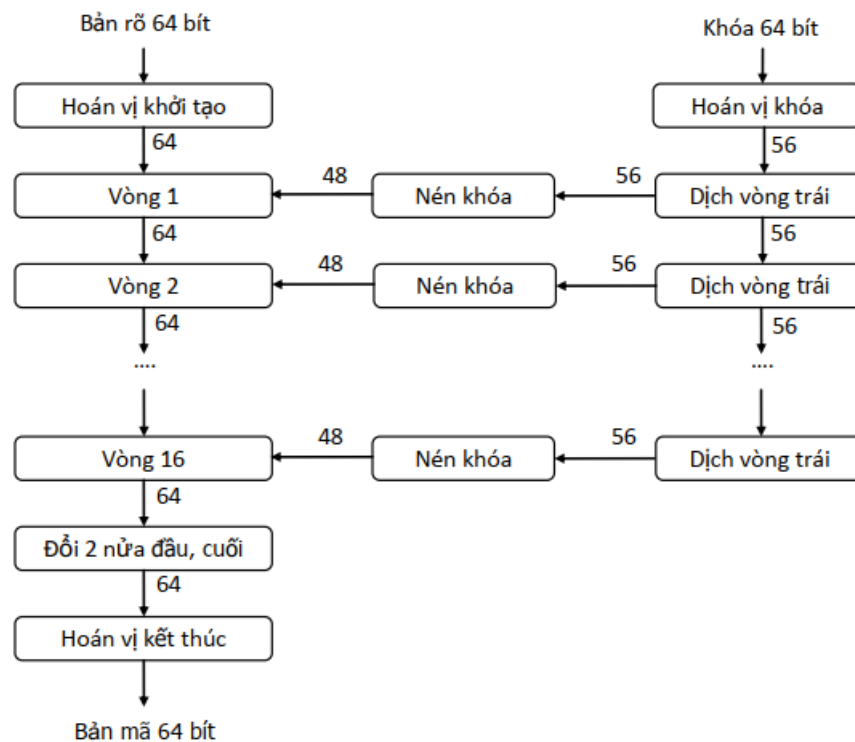
- Cài đặt và thực thi được các thuật toán mã hóa đối xứng hiện đại DES, AES bằng ngôn ngữ lập trình C# .
- Cài đặt được một chương trình mã hóa file sử dụng thuật toán mã hóa DES, AES bằng ngôn ngữ lập trình C# .
- Đánh giá độ phức tạp và độ an toàn của các thuật toán mã hóa đối xứng hiện đại DES, AES.
- Thực hiện các chiến lược nhằm kết hợp các phương pháp mã hóa trong thực tế, nâng cao tính bảo mật cho bài toán.

## B. NỘI DUNG

### 1.1. Thuật toán mã hóa DES

#### ➤ Tóm tắt lý thuyết

- Là hệ mã thuộc Feistel gồm 16 vòng, ngoài ra DES có thêm một hoán vị khởi tạo trước khi vào vòng 1 và một hoán vị khởi tạo sau vòng 16
- Kích thước của khối là 64 bit: ví dụ bản tin “*meetmeafterthetogaparty*” biểu diễn theo mã ASCII thì mã DES sẽ mã hóa làm 3 lần, mỗi lần 8 chữ cái (64 bit): *meetmeaf - tertheto - gaparty*.
- Kích thước khóa là 56 bit
- Mỗi vòng của DES dùng khóa con có kích thước 48 bit được trích ra từ khóa chính.
- Sơ đồ mã hóa DES: gồm ba phần, phần thứ nhất là các hoán vị khởi tạo và hoán vị kết thúc. Phần thứ hai là các vòng Feistel, phần thứ ba là thuật toán sinh khóa con.



- **Hoán vị khởi tạo và hoán vị kết thúc:** Đánh số các bit của khối 64 bit theo thứ tự từ trái sang phải là 0, 1, ..., 62, 63:  $b_0, b_1, \dots, b_{63}$ .

Hoán vị khởi tạo sẽ hoán đổi các bit theo quy tắc sau:

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
56	48	40	32	24	16	8	0
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6

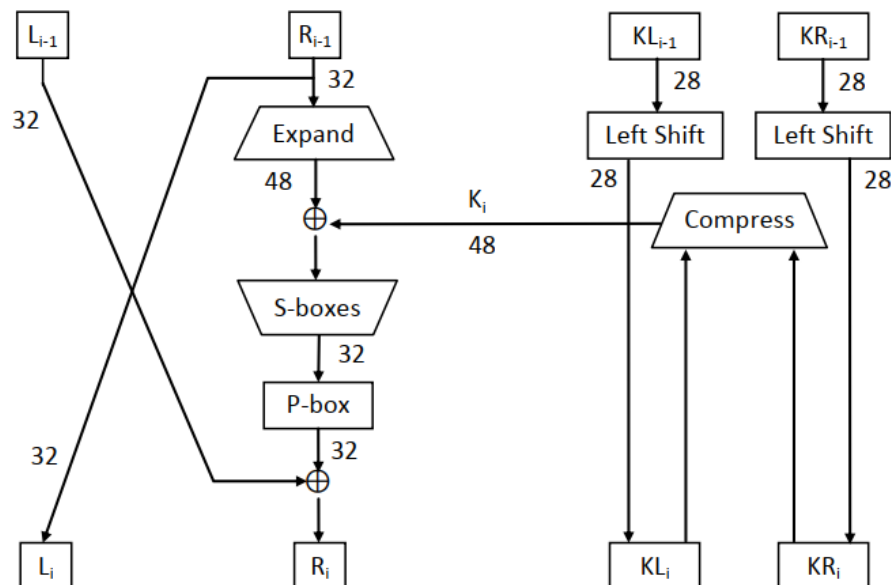
$$(b_0 b_1 b_2 \dots b_{62} b_{63} \rightarrow b_{57} b_{49} b_{41} \dots b_{14} b_6)$$

Hoán vị kết thúc hoán đổi các bit theo quy tắc sau:

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
32	0	40	8	48	16	56	24

Hoán vị kết thúc chính là hoán vị nghịch đảo của hoán vị khởi tạo

#### - Các vòng của DES



Trong DES, hàm F của Feistel là:  $F(R_{i-1}, K_i) = P\text{-box}(S\text{-boxes}(Expand(R_{i-1}) K_i))$

Hàm *Expand* vừa mở rộng vừa hoán vị  $R_{i-1}$  từ 32 bit lên 48 bit. Hàm *Sboxes* nén 48 bit lại còn 32 bit.

#### - Thuật toán sinh khóa con của DES

Khóa  $K$  64 bit ban đầu được rút trích và hoán vị thành một khóa 56 bit theo quy tắc:

56	48	40	32	24	16	8
0	57	49	41	33	25	17
9	1	58	50	42	34	26
18	10	2	59	51	43	35
62	54	46	38	30	22	14
6	61	53	45	37	29	21
13	5	60	52	44	36	28
20	12	4	27	19	11	3

Khóa 56 bit này được chia thành 2 nửa trái phải  $KL_0$  và  $KR_0$ , mỗi nửa có kích thước 28 bit. Tại vòng thứ  $i$  ( $i = 1, 2, 3, \dots, 16$ ),  $KL_{i-1}$  và  $KR_{i-1}$  được dịch vòng trái  $r_i$  bit để có được  $KL_i$  và  $KR_i$ , với  $r_i$  được định nghĩa:

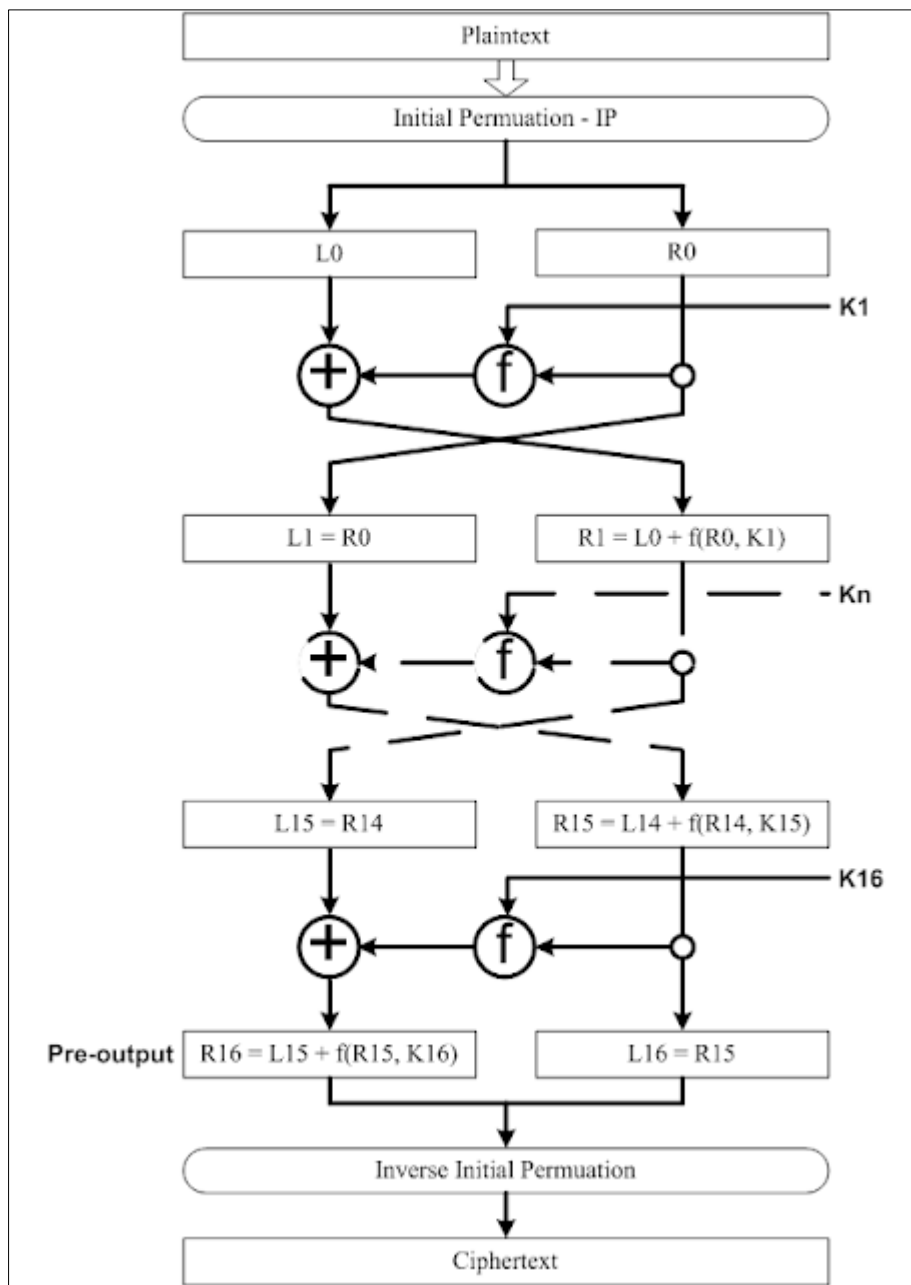
$$r_i = \begin{cases} 1 & \text{nếu } i \in \{1, 2, 9, 16\} \\ 2 & \text{với những } i \text{ khác} \end{cases}$$

Cuối cùng khóa  $K_i$  của mỗi vòng được tạo ra bằng cách hoán vị và nén 56 bit của  $KL_i$  và  $KR_i$  thành 48 bit theo quy tắc:

13	16	10	23	0	4	2	27
14	5	20	9	22	18	11	3
25	7	15	6	26	19	12	1
40	51	30	36	46	54	29	39
50	44	32	47	43	48	38	55
33	52	45	41	49	35	28	31

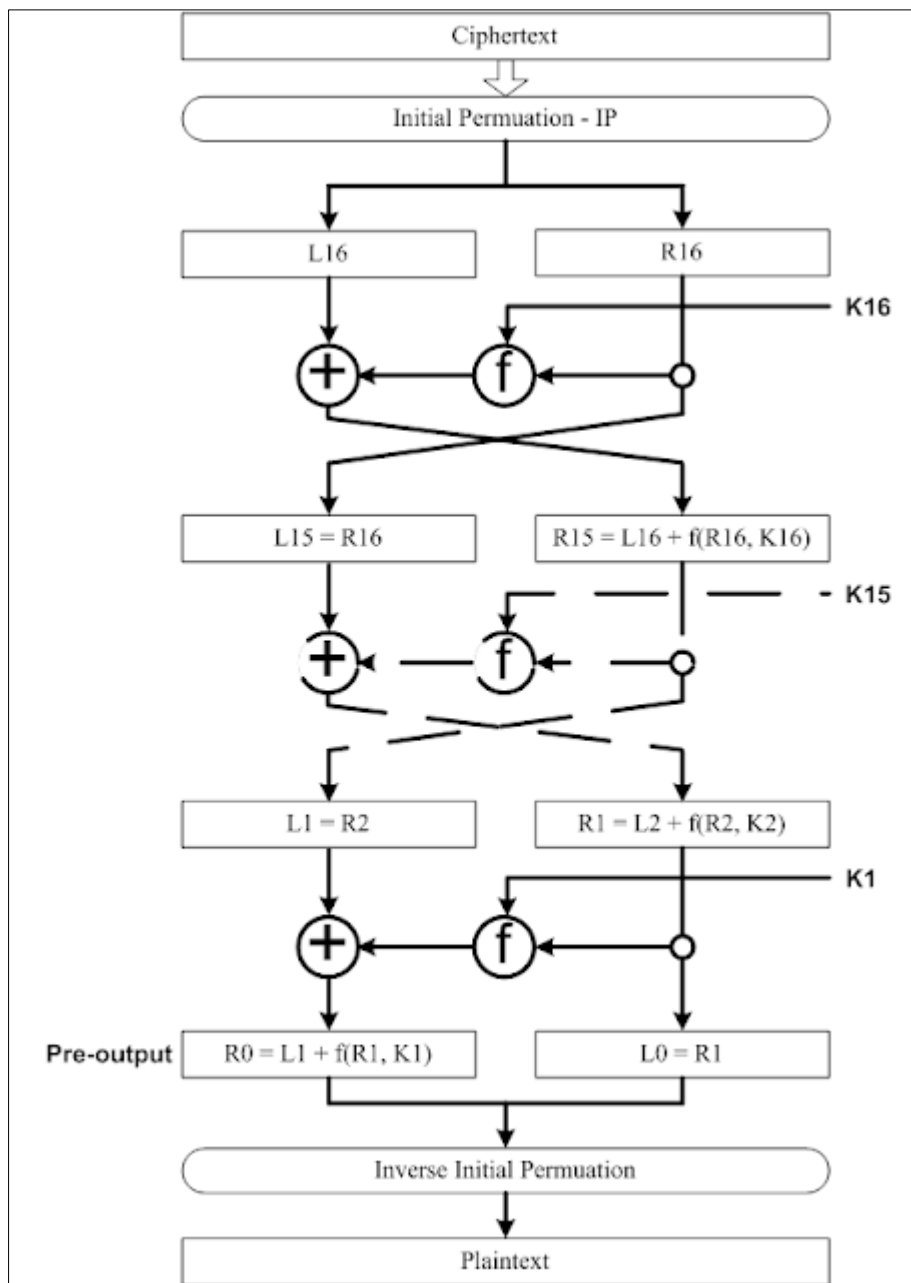
48 bit

### ➤ Sơ đồ thuật toán mã hóa DES



➤ **Thuật toán giải mã**

- Đầu vào: Dữ liệu cần giải mã (ciphertext)
- Đầu ra: Kết quả giải mã được (plaintext).
- Khóa vòng sử dụng trong các vòng lặp giải mã có thứ tự ngược với quá trình mã hóa. Tại vòng lặp giải mã đầu tiên, khóa vòng được sử dụng là K16. Tại vòng lặp giải mã thứ 2, khóa vòng được sử dụng là K15, tại vòng lặp giải mã cuối cùng thì khóa vòng được sử dụng là K1.



➤ Mã giả thuật toán DES

**Tạo 16 khóa con**

```

C[0]D[0] = PC-1 (KEY)
for i = 1 to 16
    C[i] = LeftShift[i] (C[i-1])
    D[i] = LeftShift[i] (D[i-1])
    K[i] = PC-2 (C[i]D[i])
end for

```

**Mã hóa khối dữ liệu**

```

L[0]R[0] = IP(plain block)
for i=1 to 16
    L[i] = R[i-1]

```

```

        R[i] = L[i-1] XOR F(R[i-1], K[i])
    end for
    cipher block = FP(R[16]L[16])

```

### ***Giải mã khối dữ liệu***

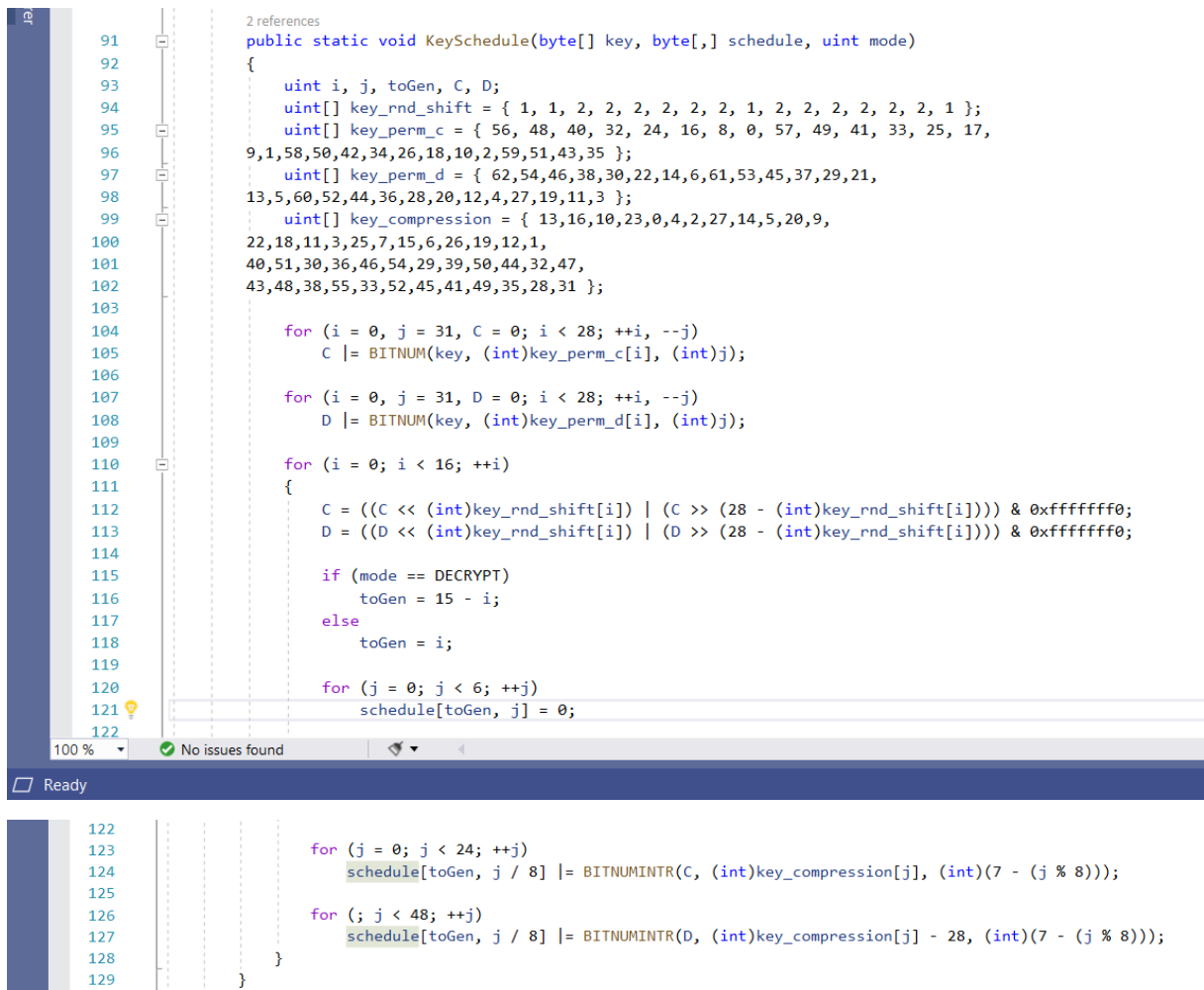
```

R[16]L[16] = IP(cipher block)
for i=1 to 16
    R[i-1] = L[i]
    L[i-1] = R[i] xor f(L[i], K[i])
end for
plain block = FP(L[0]R[0])

```

- **Cài đặt:** Thực hiện mã hóa theo thuật toán mã hóa DES với chuỗi Plaintext bất kỳ

#### **B1: Tạo khóa**



```

2 references
public static void KeySchedule(byte[] key, byte[,] schedule, uint mode)
{
    uint i, j, toGen, C, D;
    uint[] key_rnd_shift = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };
    uint[] key_perm_c = { 56, 48, 40, 32, 24, 16, 8, 0, 57, 49, 41, 33, 25, 17,
9,1,58,50,42,34,26,18,10,2,59,51,43,35 };
    uint[] key_perm_d = { 62,54,46,38,30,22,14,6,61,53,45,37,29,21,
13,5,60,52,44,36,28,20,12,4,27,19,11,3 };
    uint[] key_compression = { 13,16,10,23,0,4,2,27,14,5,20,9,
22,18,11,3,25,7,15,6,26,19,12,1,
40,51,30,36,46,54,29,39,50,44,32,47,
43,48,38,55,33,52,45,41,49,35,28,31 };

    for (i = 0, j = 31, C = 0; i < 28; ++i, --j)
        C |= BITNUM(key, (int)key_perm_c[i], (int)j);

    for (i = 0, j = 31, D = 0; i < 28; ++i, --j)
        D |= BITNUM(key, (int)key_perm_d[i], (int)j);

    for (i = 0; i < 16; ++i)
    {
        C = ((C << (int)key_rnd_shift[i]) | (C >> (28 - (int)key_rnd_shift[i]))) & 0xffffffff;
        D = ((D << (int)key_rnd_shift[i]) | (D >> (28 - (int)key_rnd_shift[i]))) & 0xffffffff;

        if (mode == DECRYPT)
            toGen = 15 - i;
        else
            toGen = i;

        for (j = 0; j < 6; ++j)
            schedule[toGen, j] = 0;

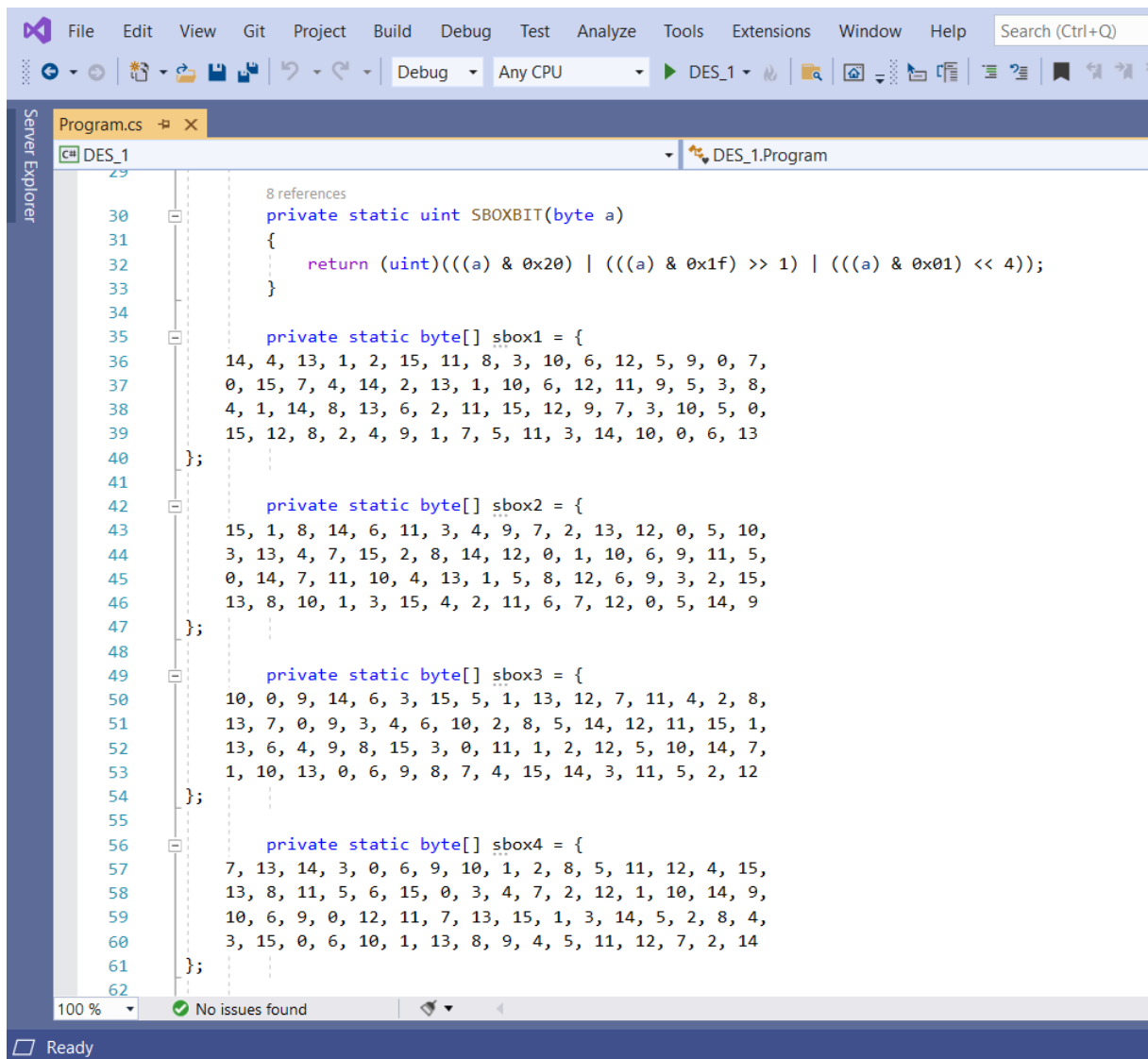
        for (j = 0; j < 24; ++j)
            schedule[toGen, j / 8] |= BITNUMINTR(C, (int)key_compression[j], (int)(7 - (j % 8)));

        for (; j < 48; ++j)
            schedule[toGen, j / 8] |= BITNUMINTR(D, (int)key_compression[j] - 28, (int)(7 - (j % 8)));
    }
}

```

#### **B2: Tạo S-Box**





Visual Studio interface showing a C# program for DES encryption. The code defines four S-boxes (SBOXBIT and sbox1-sbox4) used for bit manipulation. The interface includes a menu bar, toolbar, and a status bar at the bottom.

```
29
30 private static uint SBOXBIT(byte a)
31 {
32     return (uint)((((a) & 0x20) | (((a) & 0x1f) >> 1) | (((a) & 0x01) << 4)));
33 }
34
35 private static byte[] sbox1 = {
36     14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
37     0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
38     4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
39     15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13
40 };
41
42 private static byte[] sbox2 = {
43     15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
44     3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
45     0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
46     13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9
47 };
48
49 private static byte[] sbox3 = {
50     10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
51     13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
52     13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
53     1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12
54 };
55
56 private static byte[] sbox4 = {
57     7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
58     13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
59     10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
60     3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14
61 };
62
```

8 references

100 % No issues found

Ready

```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)
Debug Any CPU DES_1
Server Explorer Program.cs DES_1
DES_1
62
63 private static byte[] sbox5 = {
64     2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
65     14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
66     4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
67     11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3
68 };
69
70 private static byte[] sbox6 = {
71     12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
72     10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
73     9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
74     4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13
75 };
76
77 private static byte[] sbox7 = {
78     4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
79     13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
80     1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
81     6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12
82 };
83
84 private static byte[] sbox8 = {
85     13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
86     1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
87     7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
88     2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11
89 };
90

```

### B3: Hàm mã hóa

```

240 public static void Crypt(byte[] input, byte[] output, byte[][] key)
241 {
242     uint[] state = new uint[2];
243     uint idx, t;
244
245     IP(state, input);
246
247     for (idx = 0; idx < 15; ++idx)
248     {
249         t = state[1];
250         state[1] = F(state[1], key[idx]) ^ state[0];
251         state[0] = t;
252     }
253
254     state[0] = F(state[1], key[15]) ^ state[0];
255
256     InvIP(state, output);
257 }

```

### Kết quả

```

Microsoft Visual Studio Debug Console
Encrypt Output: c9 57 44 25 6a 5e d3 1d
Decrypt Output: 01 23 45 67 89 ab cd e7

```

## 1.2. Thuật toán mã hóa AES

### ➤ Tóm tắt lý thuyết

- AES làm việc với các khối dữ liệu 128bit và độ dài khóa 128bit, 192bit hoặc 256bit.
- Các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu đầu vào 128bit được chia thành 16byte, có thể xếp thành 4 cột, mỗi cột 4 phần tử hay một ma trận 4x4 của các byte, nó gọi là ma trận trạng thái.
- Tùy thuộc vào độ dài của khóa khi sử dụng 128bit, 192bit hay 256bit mà thuật toán được thực hiện với số lần lặp khác nhau.

### ➤ Thuật toán

#### **Xây dựng bảng S-box.**

##### **a. Bảng S – box thuận, S-Box nghịch**

- Bảng S-box thuận được sinh ra bằng việc xác định nghịch đảo cho một giá trị nhất định trên  $GF(2^8) = GF(2)[x] / (x^8 + x^4 + x^3 + x + 1)$ . Giá trị 0 không có nghịch đảo thì được ánh xạ với 0. Những nghịch đảo được chuyển đổi thông qua phép biến đổi affine.
- S-box nghịch đảo chỉ đơn giản là S-box chạy ngược. Nó được tính bằng phép biến đổi affine nghịch đảo các giá trị đầu vào.

##### **b. Thuật toán sinh khóa phụ**

- Rotword: quay trái 8 bit
- SubBytes
- Tính giá trị Rcon(i) Trong đó:  $Rcon(i) = x^{(i-1)} \bmod (x^8 + x^4 + x^3 + x + 1)$
- ShiftRow

##### **c. Thuật toán mã hoá**

- Hàm AddRoundKey

```
public State addRoundKey(Key key, int round)
{
    State s = new State();
    for (int c = 0; c < nrofCol; c++)
    {
        for (int r = 0; r < nrofRow; r++)
        {
            s.buf[r, c] = (byte) (buf[r, c] ^ key.w[r, c +
4 * round]);
            //Console.Out.WriteLine("state[" + r + ", " + c
+ "]= " + buf[r, c] + " XOR " + key.w[r, c + 4 * round] + " = " +
s.buf[r, c]);
        }
    }
    return (s);
}
```

- **Hàm SubBytes**

```
public State subBytes()
{
    State s = new State();

    for (int i = 0; i < nrofRow; i++)
    {
        for (int j = 0; j < nrofCol; j++)
        {
            s.buf[i, j] = Sbox.sbox[buf[i, j]];
            //Console.Out.WriteLine("state[" + i + "," + j
+ "]" = " + buf[i, j] + "--->" + s.buf[i, j]);
        }
    }
    return (s);
}
```

```
public State subBytesInv()
{
    // TODO
    State s = new State();
    for (int i = 0; i < nrofRow; i++)
    {
        for (int j = 0; j < nrofCol; j++)
        {
            s.buf[i, j] = Sbox.sboxInv[buf[i, j]];
            //Console.Out.WriteLine("state[" + i + "," + j
+ "]" = " + buf[i, j] + "--->" + s.buf[i, j]);
        }
    }
    return (s);
}
```

- **Hàm ShiftRow**

```
public State shiftRows()
{
    State s = new State();

    for (int i = 0; i < nrofRow; i++)
    {
        for (int j = 0; j < nrofCol; j++)
        {
            s.buf[i, j] = buf[i, (i + j) % 4];
            //Console.Out.WriteLine("state[" + i + "," + j
+ "]" = " + s.buf[i, j]);
        }
    }
    return (s);
}
```

```
public State shiftRowsInv()
{
    State s = new State();

    for (int i = 0; i < nrofRow; i++)
```

```

        {
            for (int j = 0; j < nrofCol; j++)
            {
                s.buf[i, j] = buf[i, (j - i + 4) % 4];
                // Console.Out.WriteLine("state[" + i + ", " +
j + "]=" + s.buf[i, j]);
            }
        }
        return (s);
    }

```

- **Hàm MixColumns**

```

public State mixColumns()
{
    State s = new State();
    for (int c = 0; c < nrofCol; c++)
    {
        s.buf[0, c] = (byte) (mul2[buf[0, c]] ^ mul3[buf[1,
c]] ^ buf[2, c] ^ buf[3, c]);
        s.buf[1, c] = (byte) (buf[0, c] ^ mul2[buf[1, c]] ^
mul3[buf[2, c]] ^ buf[3, c]);
        s.buf[2, c] = (byte) (buf[0, c] ^ buf[1, c] ^
mul2[buf[2, c]] ^ mul3[buf[3, c]]);
        s.buf[3, c] = (byte) (mul3[buf[0, c]] ^ buf[1, c] ^
buf[2, c] ^ mul2[buf[3, c]]);
    }
    return (s);
}

```

```

public State mixColumnsInv()
{
    State s = new State();
    for (int c = 0; c < nrofCol; c++)
    {
        s.buf[0, c] = (byte) (mul14[buf[0, c]] ^ mul11[buf[1, c]] ^
mul13[buf[2, c]] ^ mul09[buf[3, c]]);
        s.buf[1, c] = (byte) (mul09[buf[0, c]] ^ mul14[buf[1, c]] ^
mul11[buf[2, c]] ^ mul13[buf[3, c]]);
        s.buf[2, c] = (byte) (mul13[buf[0, c]] ^ mul09[buf[1, c]] ^
mul14[buf[2, c]] ^ mul11[buf[3, c]]);
        s.buf[3, c] = (byte) (mul11[buf[0, c]] ^ mul13[buf[1, c]] ^
mul09[buf[2, c]] ^ mul14[buf[3, c]]);
    }
    return (s);
}

```

#### d. Thuật toán giải mã

```

InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
    for round = Nr-1 downto 1
        InvShiftRows(state)

```

```

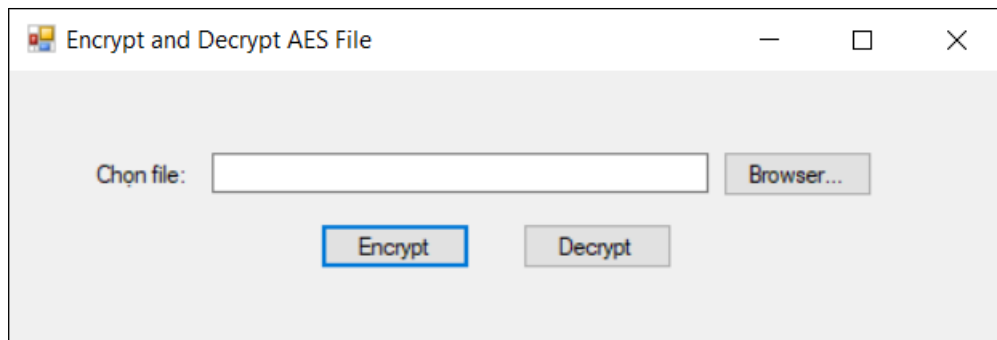
        InvSubBytes(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
        InvMixColumns(state)
    end for
    InvShiftRows(state)
    InvSubBytes(state)
    AddRoundKey(state, w[0, Nb-1])
    out = state
end

```

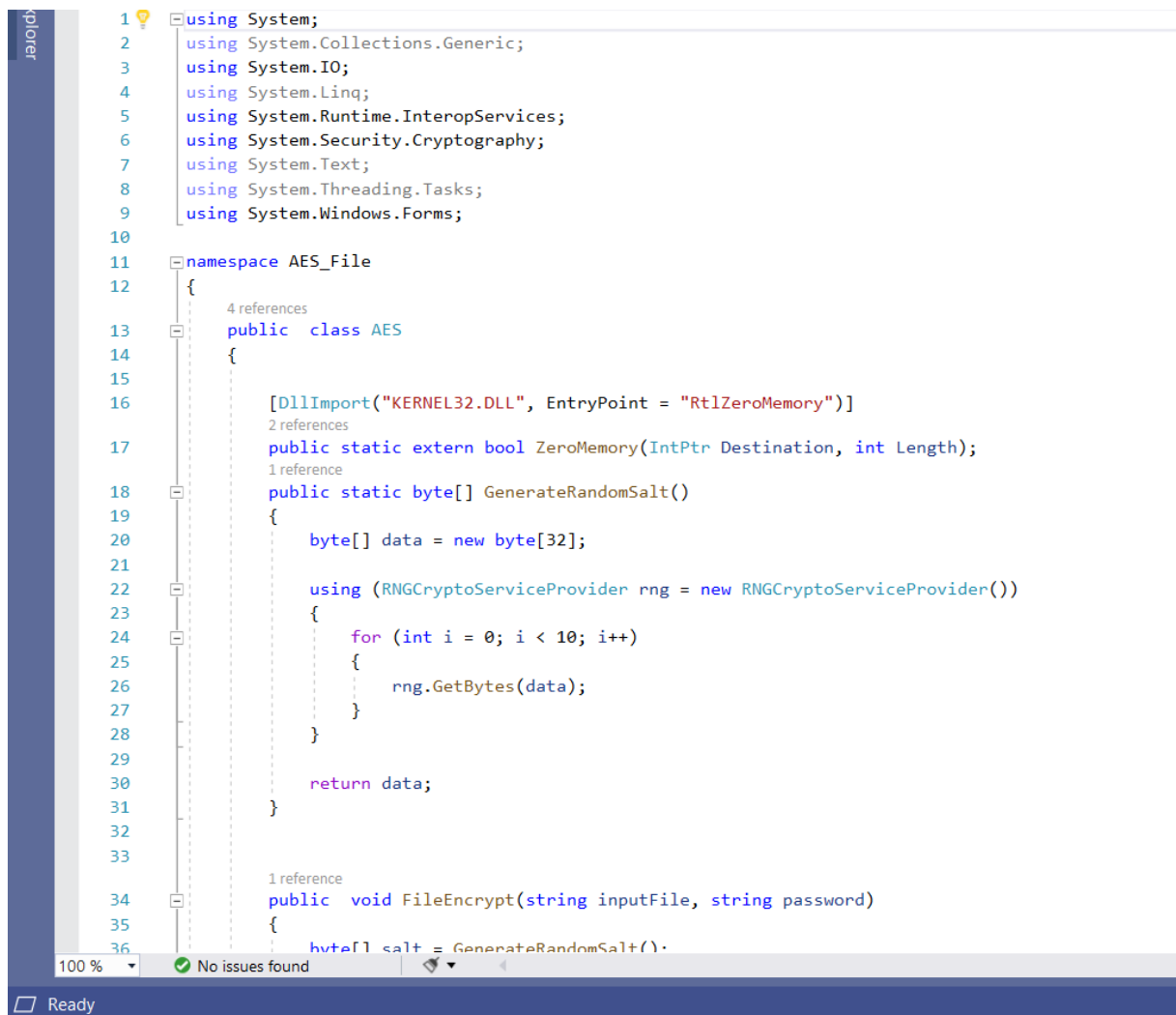
Trong đó :

- In[] : Mảng dữ liệu đầu vào Input.
- Out[] : Mảng dữ liệu đầu ra Output.
- Nr : Số vòng lặp.(Nr = 10).
- Nb : Số cột(Nb = 4).
- W[] : Mảng các w[i] có độ dài 4 bytes.

➤ **Ví dụ:** Thực hiện mã hóa một file theo thuật toán AES.



B1: Class AES



The screenshot shows a Visual Studio Code editor with a C# file named `AES_File.cs`. The code defines a namespace `AES_File` containing a `public class AES` and a `public void FileEncrypt` method. The `AES` class has two static methods: `ZeroMemory` and `GenerateRandomSalt`. The `FileEncrypt` method calls `GenerateRandomSalt`. The code is well-commented with references to other parts of the code. The status bar at the bottom indicates "No issues found" and "Ready".

```
1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using System.Linq;
5 using System.Runtime.InteropServices;
6 using System.Security.Cryptography;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace AES_File
12 {
13     4 references
14     public class AES
15     {
16         [DllImport("KERNEL32.DLL", EntryPoint = "RtlZeroMemory")]
17         2 references
18         public static extern bool ZeroMemory(IntPtr Destination, int Length);
19         1 reference
20         public static byte[] GenerateRandomSalt()
21         {
22             byte[] data = new byte[32];
23
24             using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
25             {
26                 for (int i = 0; i < 10; i++)
27                 {
28                     rng.GetBytes(data);
29                 }
30             }
31
32             return data;
33         }
34         1 reference
35         public void FileEncrypt(string inputFile, string password)
36         {
37             byte[] salt = GenerateRandomSalt();
38         }
39     }
40 }
```

## B2: Hàm mã hóa file

```
34 public void FileEncrypt(string inputFile, string password)
35 {
36     byte[] salt = GenerateRandomSalt();
37     FileStream fsCrypt = new FileStream(inputFile + ".aes", FileMode.Create);
38     byte[] passwordBytes = System.Text.Encoding.UTF8.GetBytes(password);
39     RijndaelManaged AES = new RijndaelManaged();
40     AES.KeySize = 256;
41     AES.BlockSize = 128;
42     AES.Padding = PaddingMode.PKCS7;
43     var key = new Rfc2898DeriveBytes(passwordBytes, salt, 50000);
44     AES.Key = key.GetBytes(AES.KeySize / 8);
45     AES.IV = key.GetBytes(AES.BlockSize / 8);
46     AES.Mode = CipherMode.CFB;
47     fsCrypt.Write(salt, 0, salt.Length);
48     CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateEncryptor(), CryptoStreamMode.Write);
49     FileStream fsIn = new FileStream(inputFile, FileMode.Open);
50     byte[] buffer = new byte[1048576];
51     int read;
52     try
53     {
54         while ((read = fsIn.Read(buffer, 0, buffer.Length)) > 0)
55         {
56             Application.DoEvents();
57             cs.Write(buffer, 0, read);
58         }
59         fsIn.Close();
60     }
61     catch (Exception ex)
62     {
63         Console.WriteLine("Error: " + ex.Message);
64     }
65     finally
66     {
67         cs.Close();
68         fsCrypt.Close();
69     }
70 }
```

```
34 public void FileEncrypt(string inputFile, string password)
35 {
36     byte[] salt = GenerateRandomSalt();
37
38     FileStream fsCrypt = new FileStream(inputFile + ".aes", FileMode.Create);
39
40     byte[] passwordBytes = System.Text.Encoding.UTF8.GetBytes(password);
41
42     RijndaelManaged AES = new RijndaelManaged();
43     AES.KeySize = 256;
44     AES.BlockSize = 128;
45     AES.Padding = PaddingMode.PKCS7;
46     var key = new Rfc2898DeriveBytes(passwordBytes, salt, 50000);
47     AES.Key = key.GetBytes(AES.KeySize / 8);
48     AES.IV = key.GetBytes(AES.BlockSize / 8);
49
50
51     AES.Mode = CipherMode.CFB;
52
53     fsCrypt.Write(salt, 0, salt.Length);
54
55     CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateEncryptor(), CryptoStreamMode.Write);
56
57     FileStream fsIn = new FileStream(inputFile, FileMode.Open);
58
59     byte[] buffer = new byte[1048576];
60     int read;
61
62     try
63     {
64         while ((read = fsIn.Read(buffer, 0, buffer.Length)) > 0)
65         {
66             Application.DoEvents();
67             cs.Write(buffer, 0, read);
68         }
69
70         fsIn.Close();
71     }
72     catch (Exception ex)
```

100 % No issues found

Ready



### B3: Hàm giải mã

```
72 public void FileDecrypt(string inputFile, string outputFile, string password)
73 {
74     byte[] passwordBytes = System.Text.Encoding.UTF8.GetBytes(password);
75     byte[] salt = new byte[32];
76     FileStream fsCrypt = new FileStream(inputFile, FileMode.Open);
77     fsCrypt.Read(salt, 0, salt.Length);
78     RijndaelManaged AES = new RijndaelManaged();
79     AES.KeySize = 256;
80     AES.BlockSize = 128;
81     var key = new Rfc2898DeriveBytes(passwordBytes, salt, 50000);
82     AES.Key = key.GetBytes(AES.KeySize / 8);
83     AES.IV = key.GetBytes(AES.BlockSize / 8);
84     AES.Padding = PaddingMode.PKCS7;
85     AES.Mode = CipherMode.CFB;
86     CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateDecryptor(), CryptoStreamMode.Read);
87     FileStream fsOut = new FileStream(outputFile, FileMode.Create);
88     int read;
89     byte[] buffer = new byte[1048576];
90     try
91     {
92         while ((read = cs.Read(buffer, 0, buffer.Length)) > 0)
93         {
94             Application.DoEvents();
95             fsOut.Write(buffer, 0, read);
96         }
97     }
98     catch (CryptographicException ex_CryptographicException)
99     {
100         Console.WriteLine("CryptographicException error: " + ex_CryptographicException.Message);
101     }
102     catch (Exception ex)
103     {
104         Console.WriteLine("Error: " + ex.Message);
105     }
106     try
107     {
108         cs.Close();
```

```
106     try
107     {
108         cs.Close();
109     }
110     catch (Exception ex)
111     {
112         Console.WriteLine("Error by closing CryptoStream: " + ex.Message);
113     }
114     finally
115     {
116         fsOut.Close();
117         fsCrypt.Close();
118     }
119 }
120 }
121 }
122 }
```

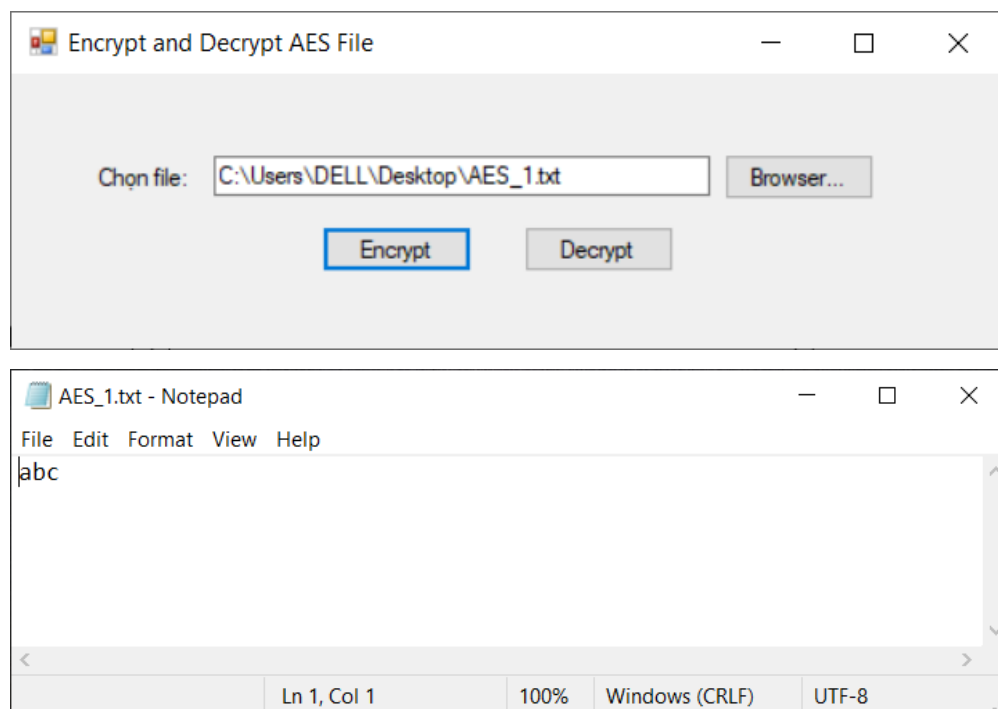
### B4: Viết các sự kiện trên Form

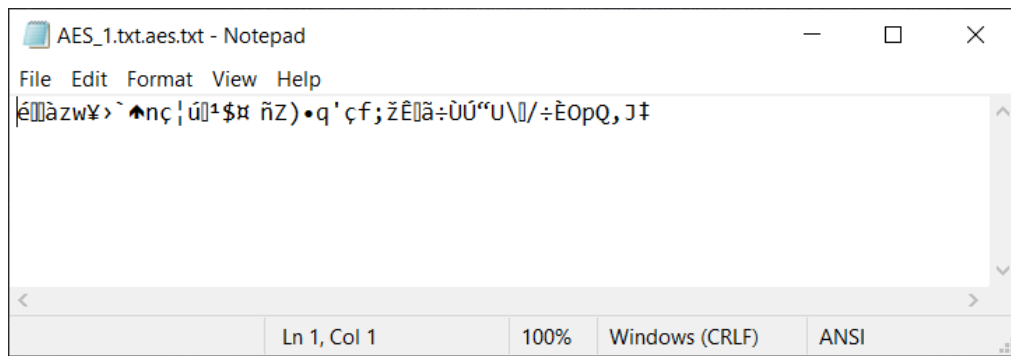
```

48 private void btn_browser_Click(object sender, EventArgs e)
49 {
50     var dlg = new OpenFileDialog();
51     if(dlg.ShowDialog() == DialogResult.OK)
52     {
53         txt_input.Text = dlg.FileName;
54     }
55 }
56
57
58
21 private void btn_encrypt_Click(object sender, EventArgs e)
22 {
23     string password = "abc";
24
25     GCHandle gch = GCHandle.Alloc(password, GCHandleType.Pinned);
26     AES.FileEncrypt(txt_input.Text, password);
27
28     AES.ZeroMemory(gch.AddrOfPinnedObject(), password.Length * 2);
29     gch.Free();
30
31     Console.WriteLine("The given password is surely nothing: " + password);
32 }
33
34 1 reference
35 private void btn_decrypt_Click(object sender, EventArgs e)
36 {
37     string password = "abc";
38
39     GCHandle gch = GCHandle.Alloc(password, GCHandleType.Pinned);
40
41     AES.FileDecrypt(txt_input.Text + ".aes", txt_input.Text, password);
42
43     AES.ZeroMemory(gch.AddrOfPinnedObject(), password.Length * 2);
44     gch.Free();
45     Console.WriteLine("The given password is surely nothing: " + password);
46 }
47

```

## Kết quả





### 1.3. Bài tập thực hành

1. Thực hiện giải mã theo thuật toán DES đã cài đặt trong **phần 1.1**
2. Viết chương trình WinForm thực hiện mã hóa file văn bản bằng thuật toán mã hóa DES, với file load bất kỳ. Các sự kiện trên form gồm: Load file, Encrypt, Decrypt.