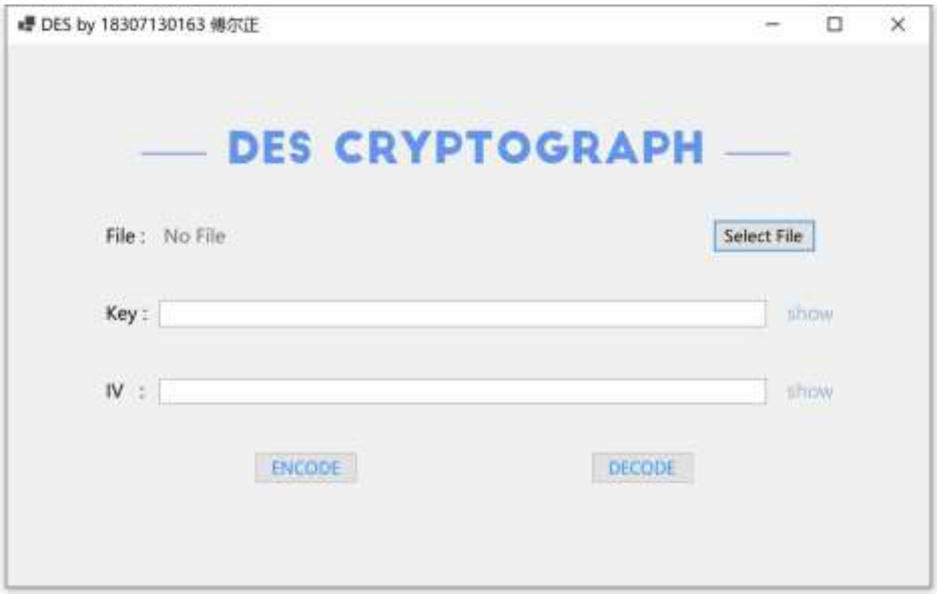
DES 加密/解密程序 说明文档 1.0

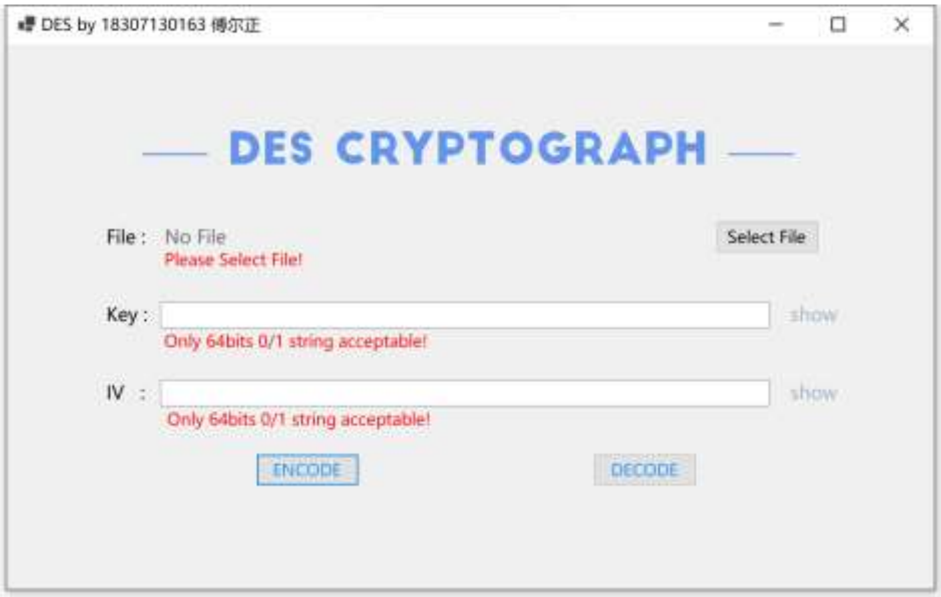
操作方法（GUI 界面）

1 加密

1. 在 Windows 操作系统下执行 `/program/DES.exe`，进入初始界面



选择加密目标文件，并输入 64 位 01 串表示主密钥，64 位 01 串表示初始向量。若输入不合法将出现以下界面

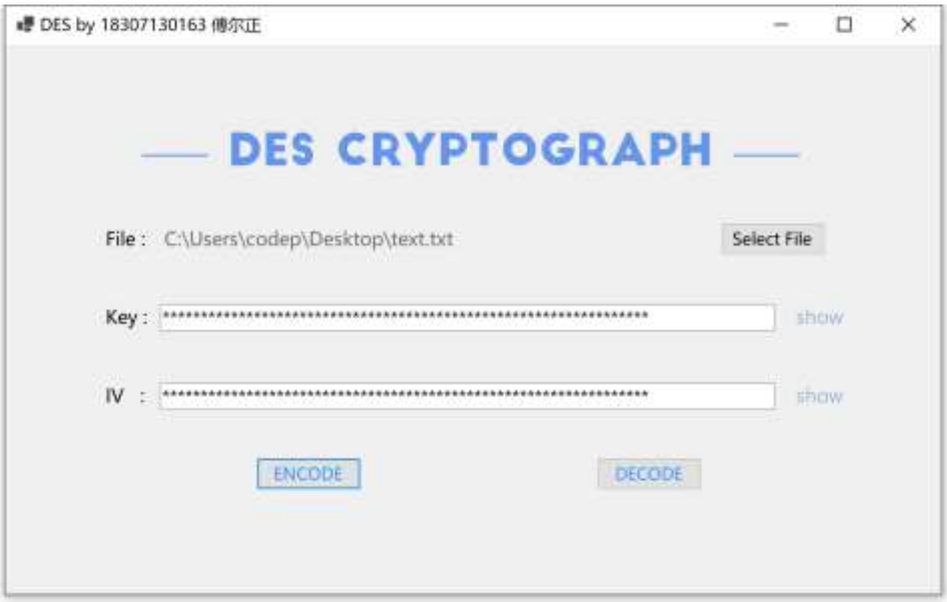


Please Select File!

Only 64bits 0/1 string acceptable!

Only 64bits 0/1 string acceptable!

被加密的文件里须保存不多于 1000 位连续的 01 字符串，若不合法发将出现异常
以下图示了一个正确的例子



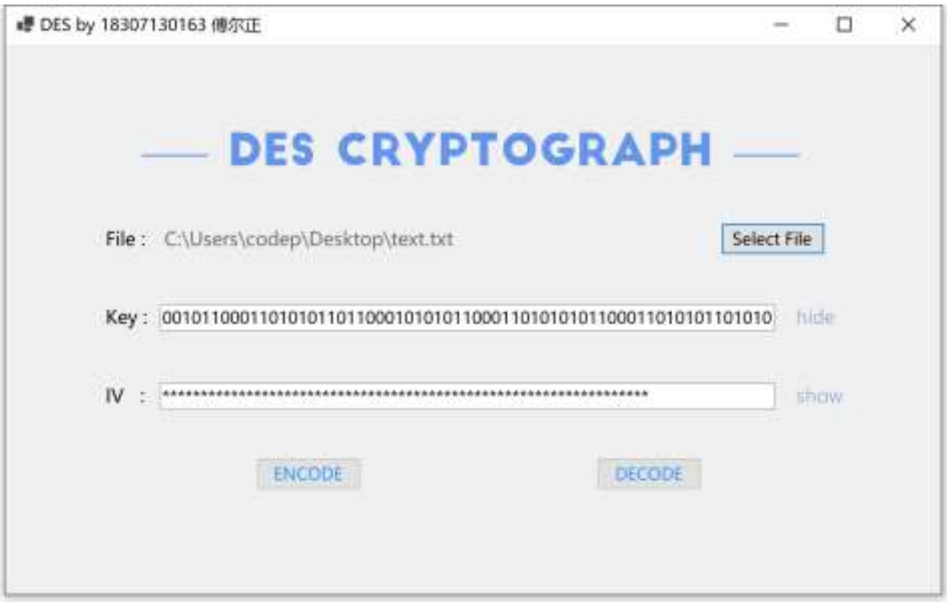
DES by 18307130163 傅尔正

File : C:\Users\codep\Desktop\text.txt

Key : **

IV : **

点击右侧的 show/hide 按钮可选择显示/隐藏文本



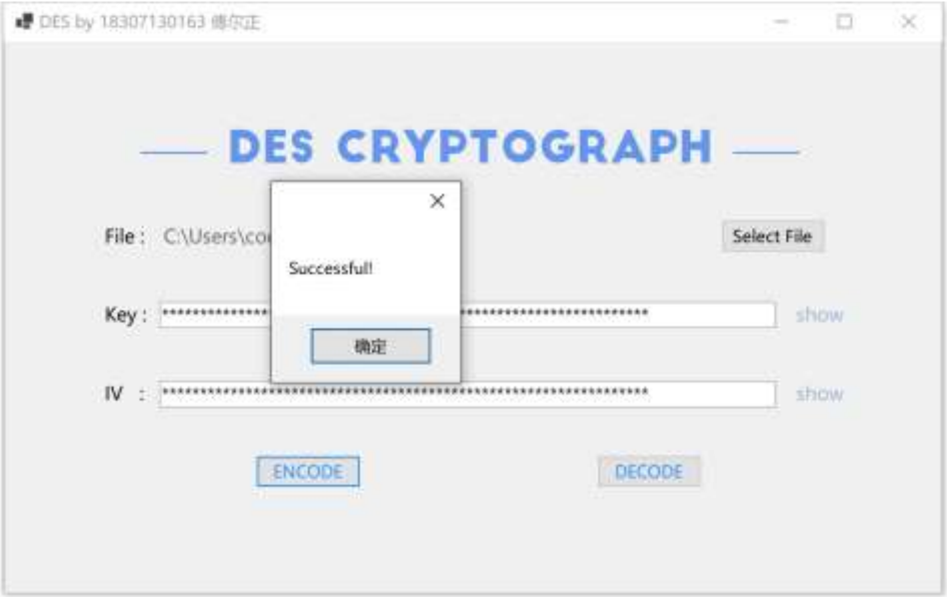
DES by 18307130163 傅尔正

File : C:\Users\codep\Desktop\text.txt

Key : 001011000110101011011000101010110001101010101100011010101101010

IV : **

点击 ENCODE 按钮，选择 TXT 文件存储路径及命名（默认为 `cipher.txt`）



DES by 18307130163 傅尔正

File : C:\Users\co

Key : **

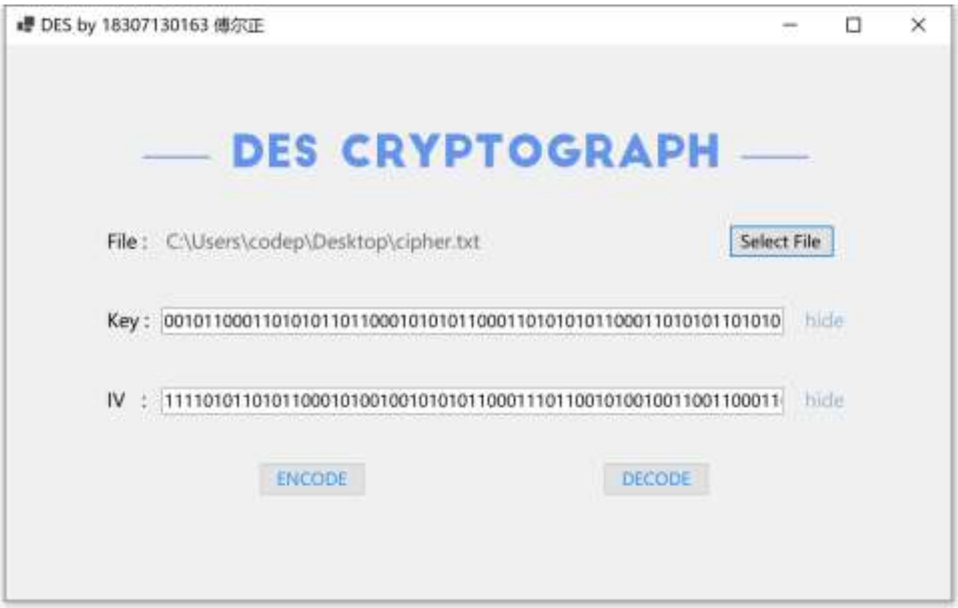
IV : **

Successful!

显示已加密成功

2 解密

解密流程与加密流程相似，解密文件须为本程序生成的密文文件，Key/IV 均须与加密所用相同

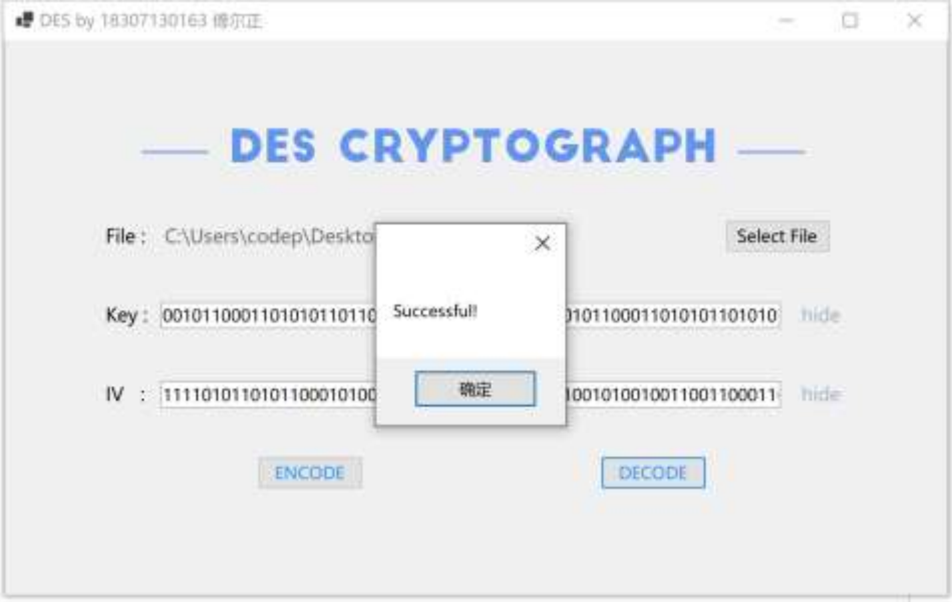


File : C:\Users\codep\Desktop\cipher.txt

Key : 00101100011010101101100010101011000110101010110001101010101101010

IV : 111101011010110001010010010101011000111011001010010011001100011

解密文件默认为 `plain.txt`



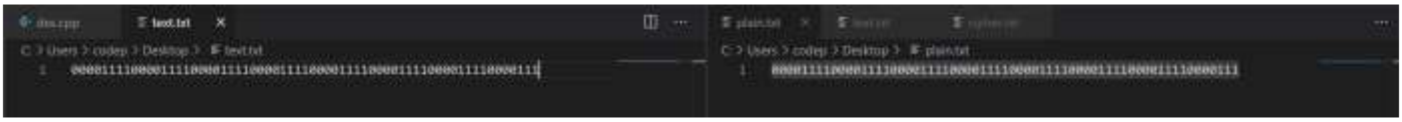
File : C:\Users\codep\Deskto

Key : 0010110001101010110110 101100011010101101010

IV : 1111010110101100010100 10010100100110011000110

Successful!

检查明文和密文，对比发现相同，说明算法实现正确



des.cpp
text.txt
C > Users > codep > Desktop > text.txt
plain.txt
text.txt
cipher.txt
C > Users > codep > Desktop > plain.txt

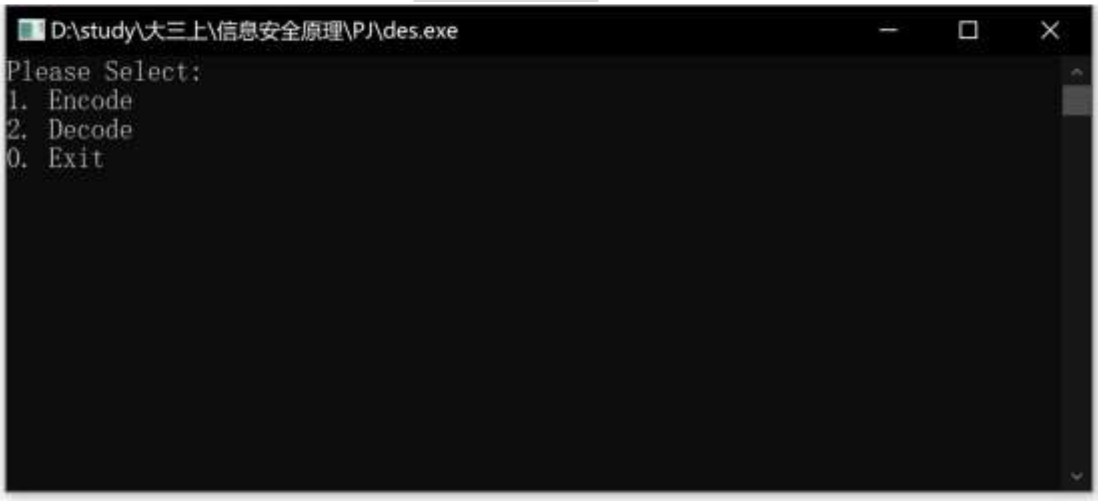
```
0000111100001111000011110000111100001111000011110000111100001111
```

```
0000111100001111000011110000111100001111000011110000111100001111
```

操作方法（CMD 命令行）

DES.exe 调用。`/program/desCore.exe`执行加密过程，而`desCore.exe`也直接提供了命令行用户交互，运行可直接使用

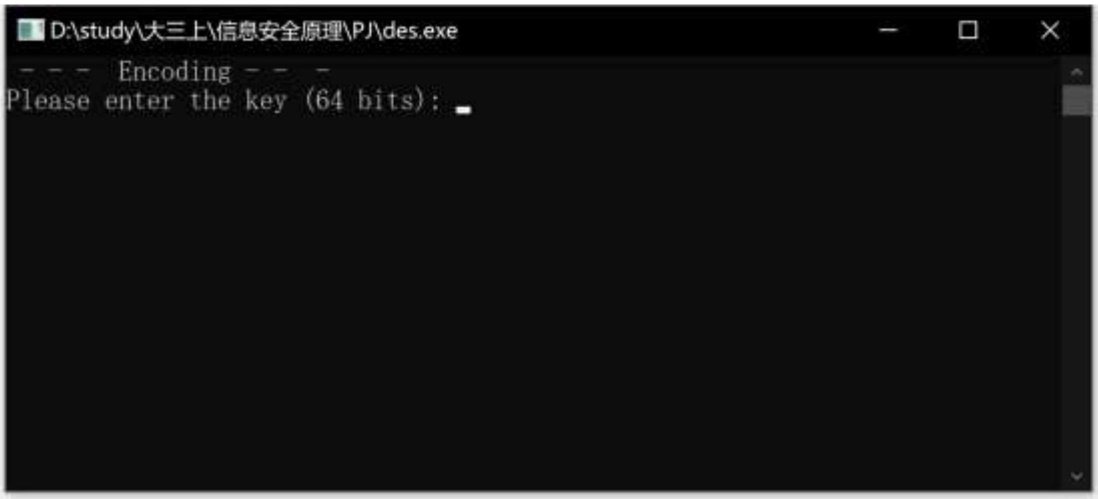
1. 在 Windows 操作系统下执行而`desCore.exe`，进入初始界面



```
Please Select:
1. Encode
2. Decode
0. Exit
```

2. 输入数字并回车选择需要进行的操作

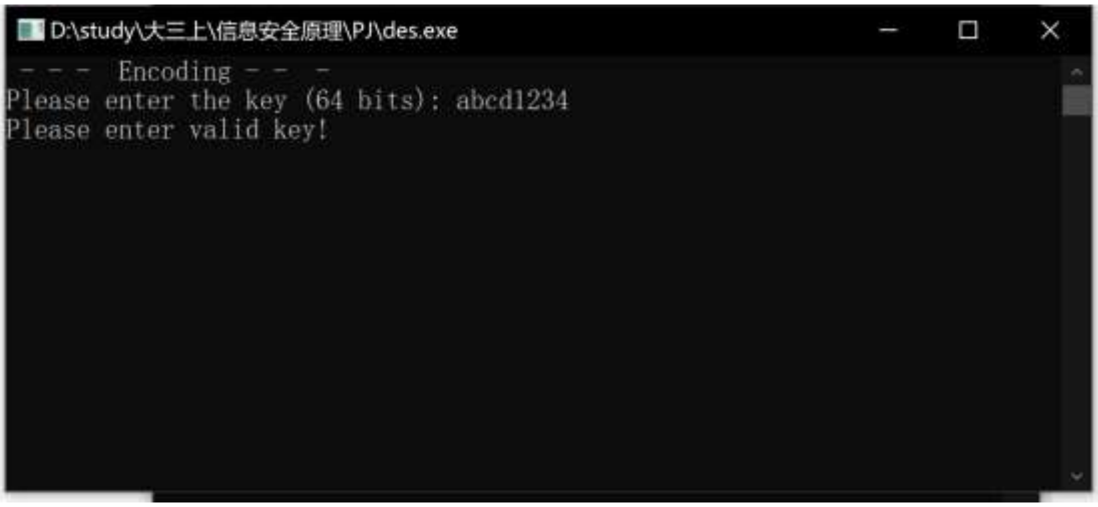
1 加密



```
 - - -  Encoding - -  -
Please enter the key (64 bits): _
```

加密界面如图示，在命令行输入 64 位 01 串表示主密钥。

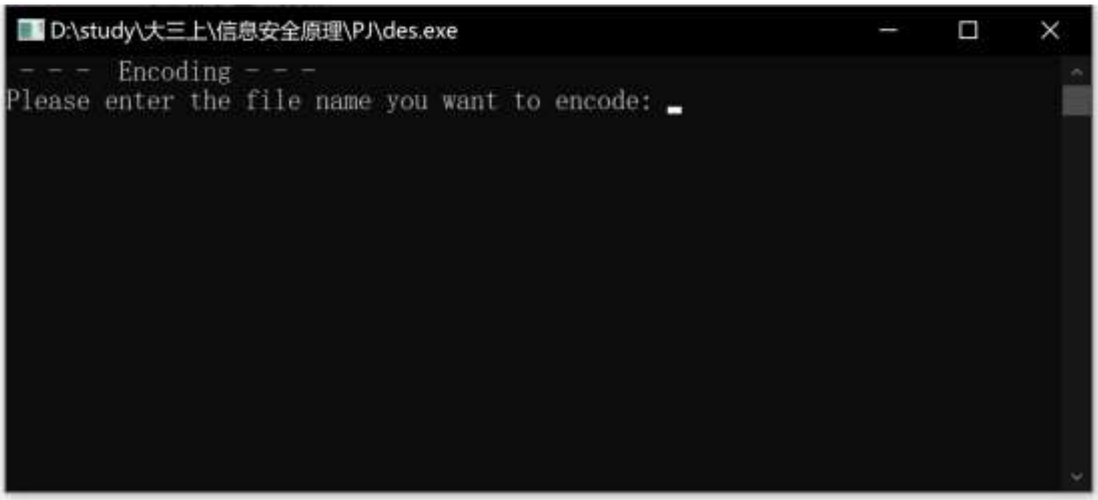
若输入不合法，将出现以下界面：



```
 - - -  Encoding - -  -
Please enter the key (64 bits): abcd1234
Please enter valid key!
```

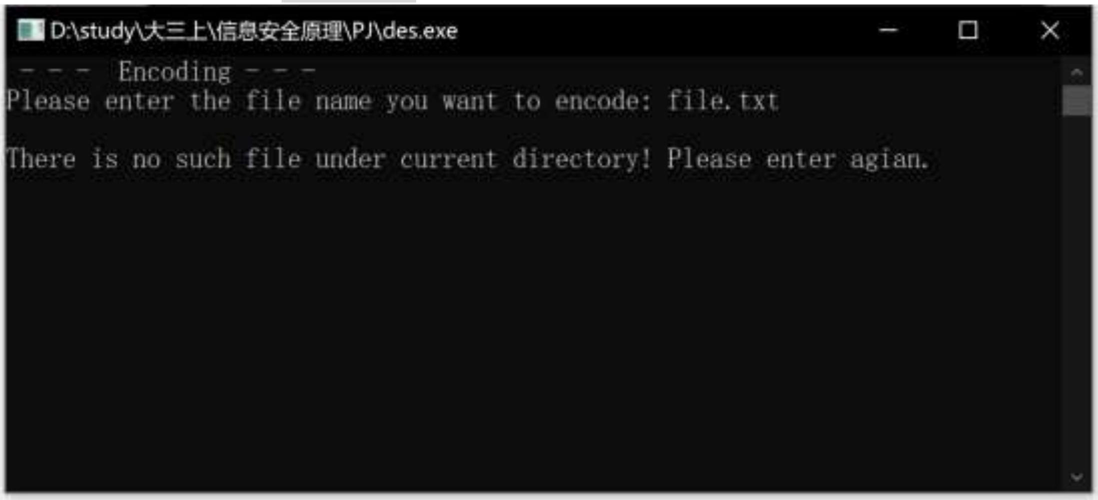
再次输入合法密钥 1111110010101001100101010111001010000101010011001010101110010100

成功进入下一流程



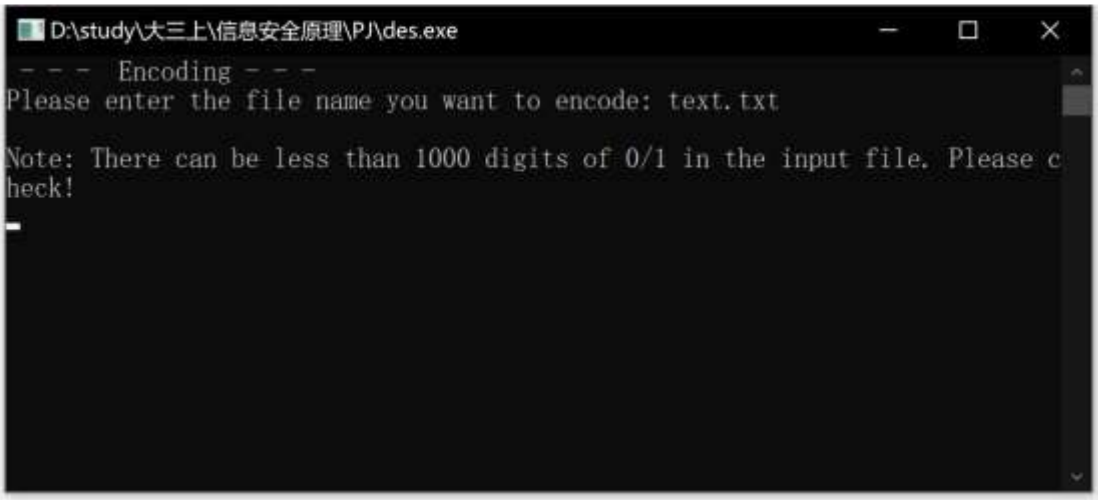
```
- - -  Encoding - - -
Please enter the file name you want to encode:
```

输入需要加密的文件名 `text.txt`，若不存在此文件，将会出现以下界面：



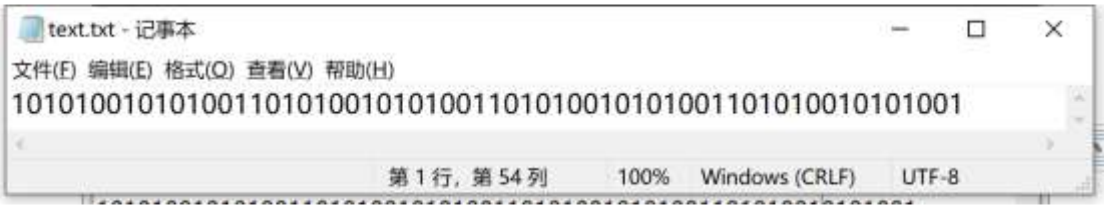
```
- - -  Encoding - - -
Please enter the file name you want to encode: file.txt

There is no such file under current directory! Please enter agian.
```

文件里须保存不多于 1000 位连续的 01 字符串，如输入不合法将出现：



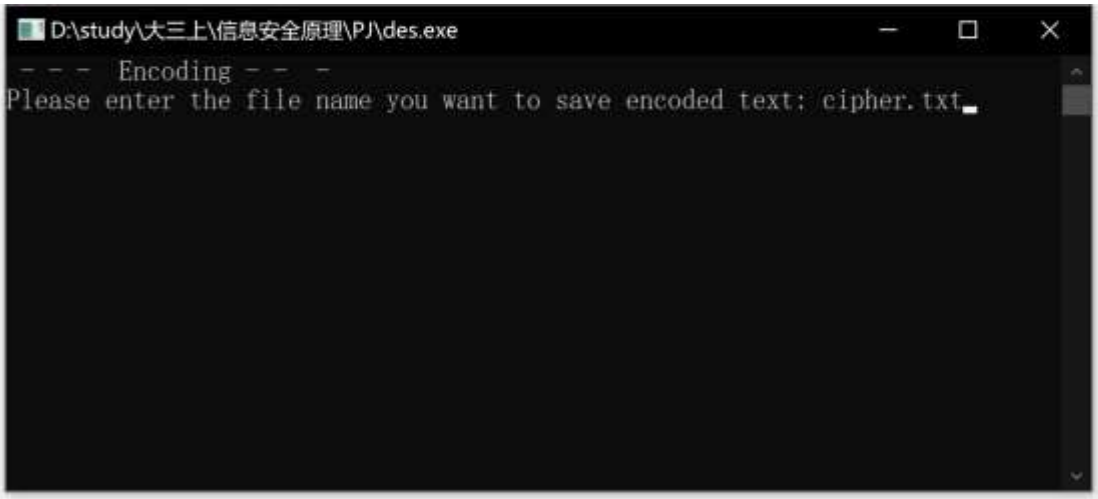
```
- - -  Encoding - - -
Please enter the file name you want to encode: text.txt

Note: There can be less than 1000 digits of 0/1 in the input file. Please c
heck!
```

检查文件后，需要加密的 01 串为



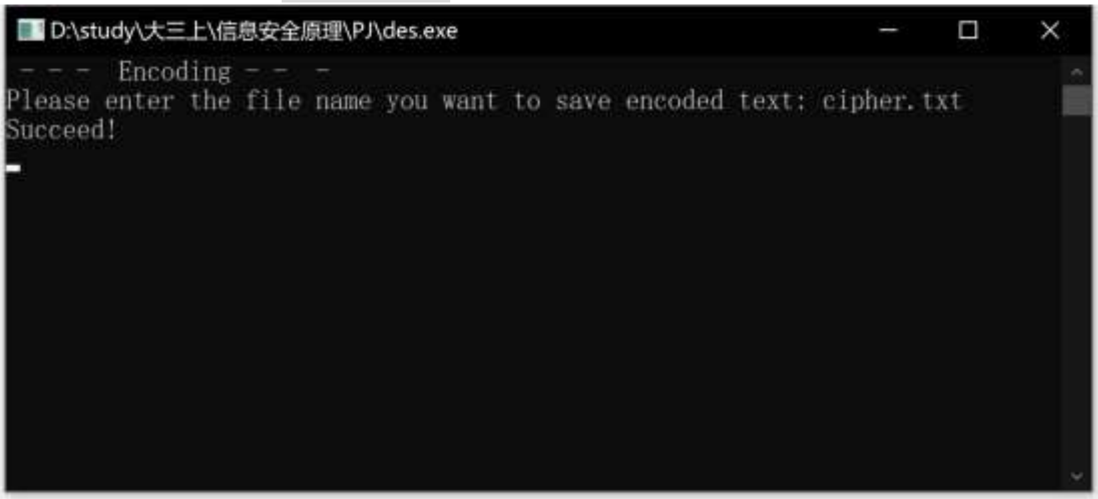
101010010101001101010010101001101010010101001101010010101001

接下来输入加密后存入的文件名



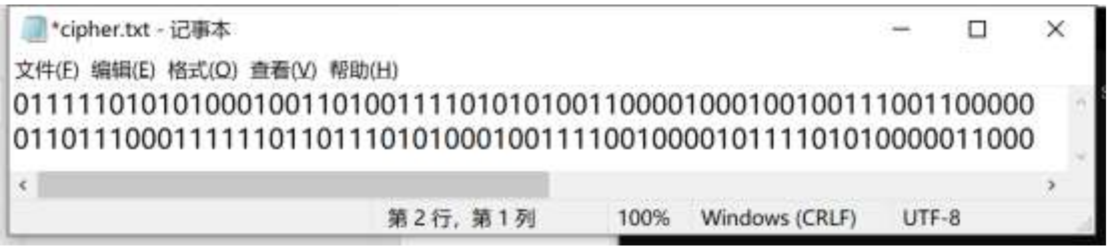
```
- - -  Encoding - -  -
Please enter the file name you want to save encoded text: cipher.txt
```

例如，将加密文本存入 `cipher.txt`



```
- - -  Encoding - -  -
Please enter the file name you want to save encoded text: cipher.txt
Succeed!
```

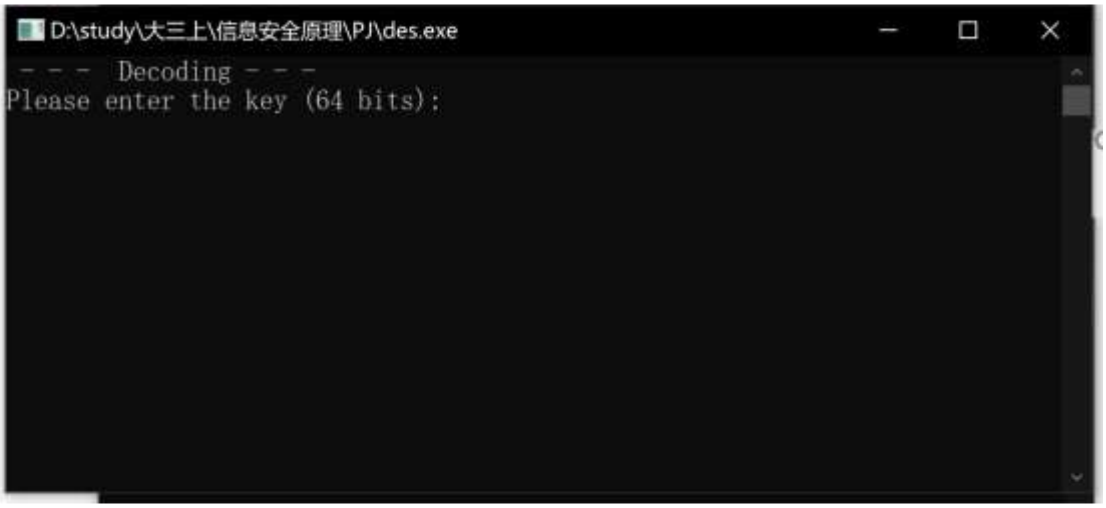
显示已成功加密，这时查看 `cipher.txt`



```
0111110101010001001101001111010101001100001000100100111001100000
0110111000111111011011101010001001111001000010111101010000011000
```

这是加密后的文本。

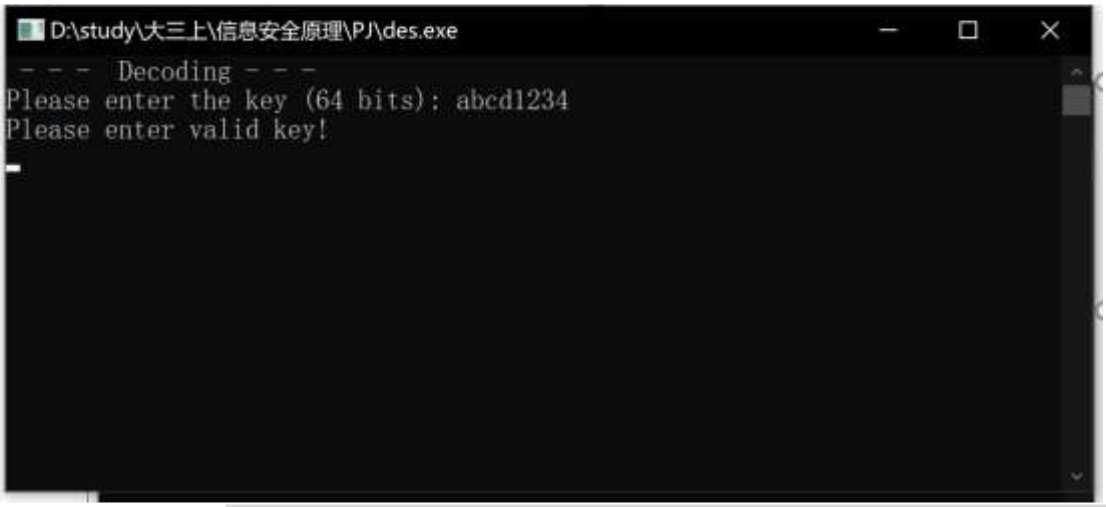
2 解密



```
- - -  Decoding - - -
Please enter the key (64 bits):
```

解密界面如图示，在命令行输入 64 位 01 串表示主密钥。

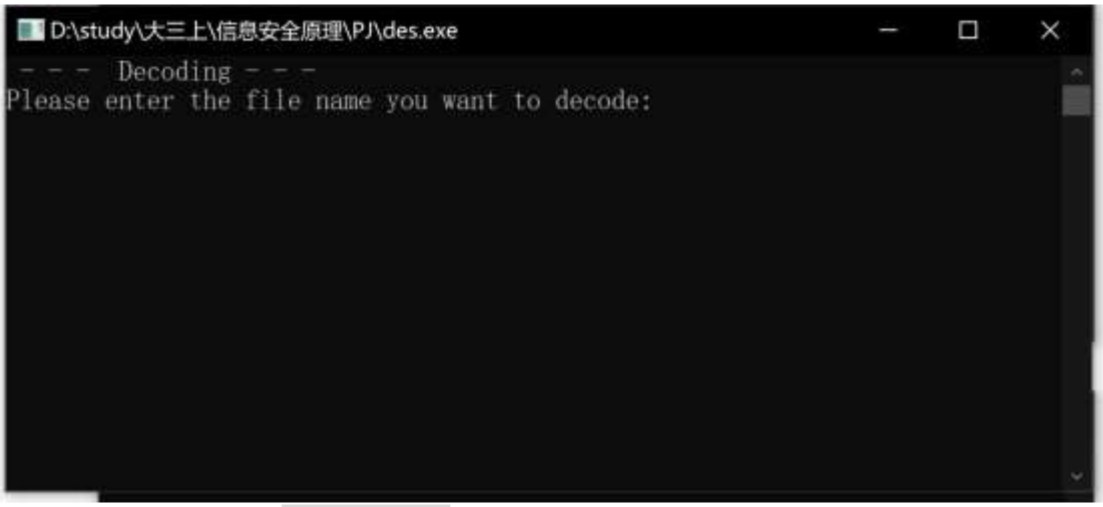
若输入不合法，将出现以下界面：



```
--- Decoding ---
Please enter the key (64 bits): abcd1234
Please enter valid key!
```

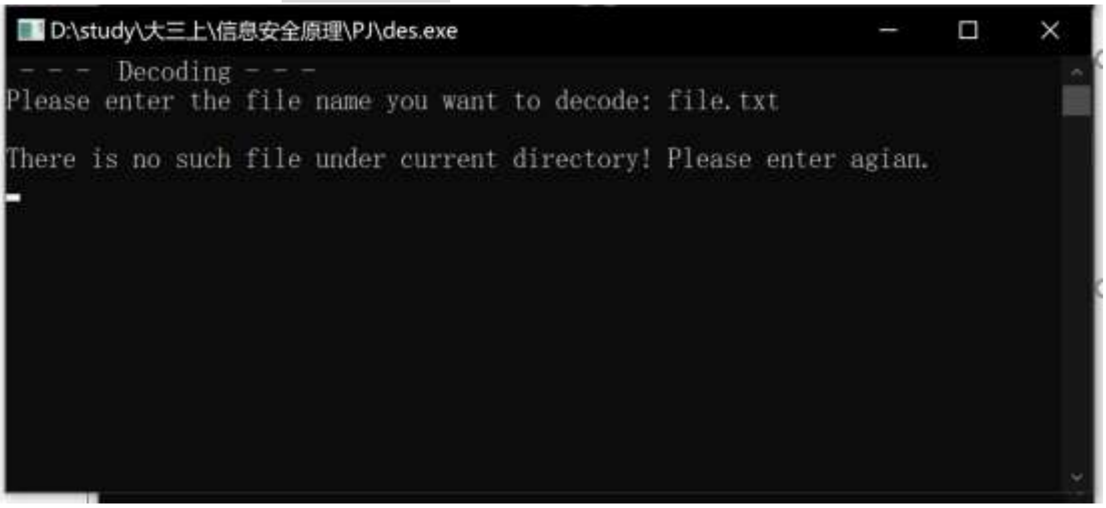
再次输入合法密钥 1111110010101001100101010111001010000101010011001010101110010100

成功进入下一流程



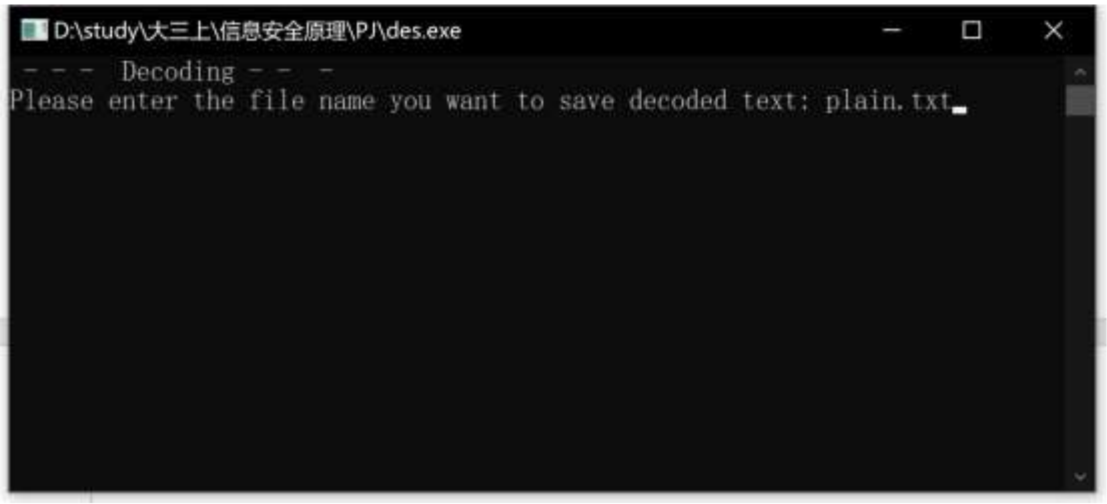
```
--- Decoding ---
Please enter the file name you want to decode:
```

输入需要解密的文件名 `cipher.txt`，若不存在此文件，将会出现以下界面：



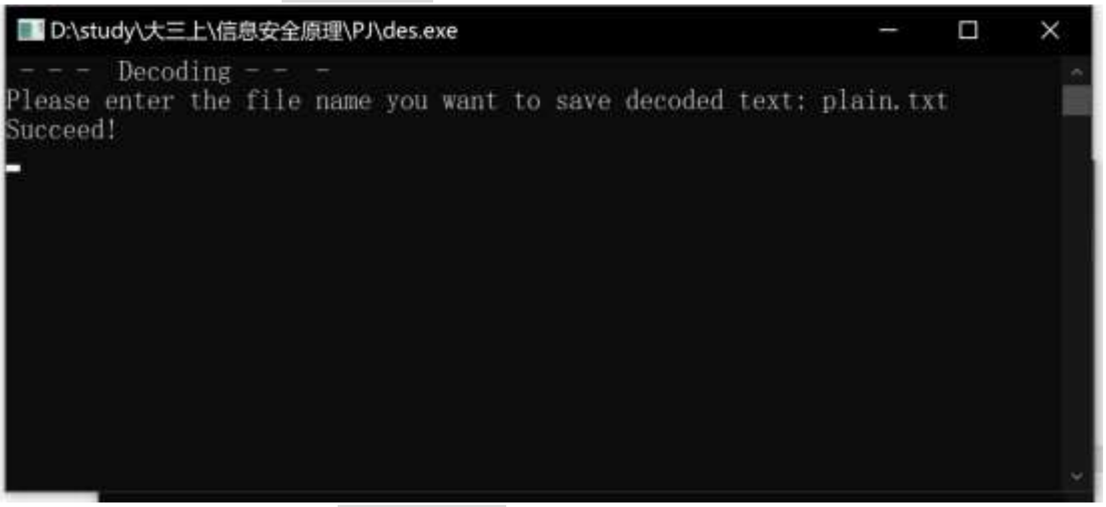
```
--- Decoding ---
Please enter the file name you want to decode: file.txt

There is no such file under current directory! Please enter agian.
```

请保证此文件由加密程序生成，成功后，接下来输入解密后存入的文件名



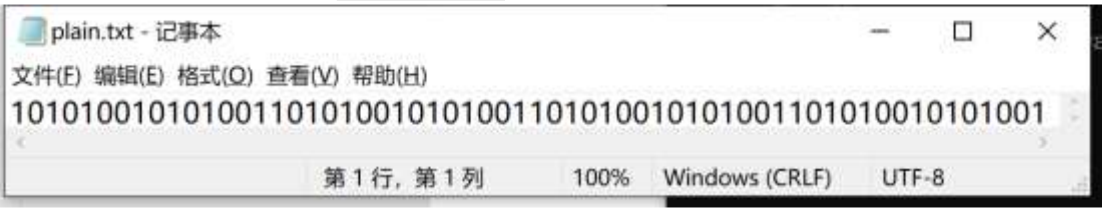
```
 - - -  Decoding - -  -
Please enter the file name you want to save decoded text: plain.txt
```

例如，将解密文本存入 `plain.txt`



```
 - - -  Decoding - -  -
Please enter the file name you want to save decoded text: plain.txt
Succeed!
```

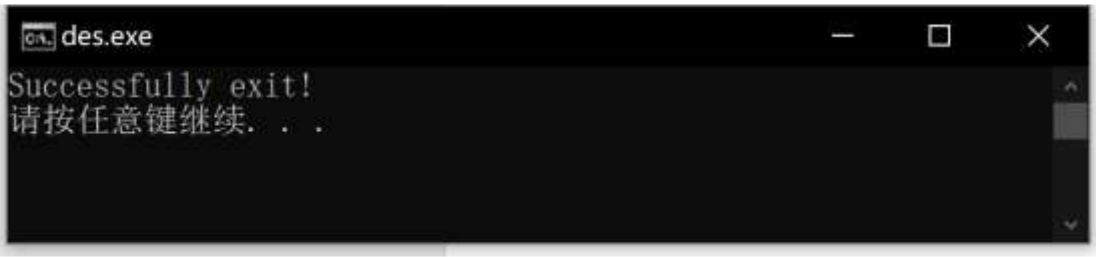
显示已成功加密，这时查看 `plaint.txt`



```
101010010101001101010010101001101010010101001101010010101001
```

发现与源文本 `text.txt` 内容相同，证明此程序正确。

0 退出



```
Successfully exit!
请按任意键继续. . .
```

成功退出

原程序解释

`des.cpp`

DES 算法主干由`/code/des.cpp`实现，namespace DES 封装了 DES 加密解密算法和输出接口，可分别通过**`DES::Encode()`**，**`DES::decode()`**和**`DES::out()`**调用。
DES 算法使用的矩阵表存入常量`IP`，`IP-1`，`IP_2`，`shift`，`E`，`S_BOX`，`P`中

- 函数**`generateKeys()`**主要功能是用主密钥生成工作密钥流

```
void generateKeys() {
    BITS56 Key;
    BITS28 left, right;
    BITS48 compressKey;
    for (int i = 0; i < 56; i++) Key[55 - i] = key[64 - PC_1[i]];
    for (int r = 0; r < 16; r++) {
        for (int i = 28; i < 56; i++) left[i - 28] = Key[i];
        for (int i = 0; i < 28; i++) right[i] = Key[i];
        Shift(left, shift[r]), Shift(right, shift[r]);
        for (int i = 28; i < 56; i++) Key[i] = left[i - 28];
        for (int i = 0; i < 28; i++) Key[i] = right[i];
        for (int i = 0; i < 48; i++) subKey[r][47 - i] = Key[56 - PC_2[i]];
    }
}
```

- 函数**`f()`**功能为计算密码函数

```
BITS32 f(BITS32 R, BITS48 K) {
    BITS48 ex = K;
    BITS32 res, tmp;
    for (int i = 0; i < 48; i++) ex[47 - i] = ex[47 - i] ^ R[32 - E[i]];
    for (int i = 0, x = 0; i < 48; i += 6, x += 4) {
        bitset<4> bi(S_BOX[i / 6][ex[47 - i] * 2 + ex[47 - i - 5]][ex[47 - i - 1] * 8 + ex[47 - i - 2] * 4 + ex[47 - i - 3] * 2 + ex[47 - i - 4]]);
        for (int j = 0; j < 4; j++) res[32 - x - j] = bi[3 - j];
    }
    tmp = res;
    for (int i = 0; i < 32; i++) res[31 - i] = tmp[32 - P[i]];
    return res;
}
```

- 函数**`Shift()`**提供了移位功能

```
void Shift(BITS28 &k, int t) {
    BITS28 res;
    for (int i = 0; i < 28; i++) res[i] = k[(i + t) % 28];
    k = res;
}
```

- **`encrypt()`**和**`decrypt()`**进行多轮加密/解密操作

```
BITS64 encrypt(BITS64 plain) {
    BITS64 res, cur;
    BITS32 left, right;
    for (int i = 0; i < 64; i++) cur[63 - i] = plain[64 - IP[i]];
    for (int i = 32; i < 64; i++) left[i - 32] = cur[i];
    for (int i = 0; i < 32; i++) right[i] = cur[i];
    for (int r = 0; r < 16; r++) {
        BITS32 tmp = right;
        right = left ^ f(right, subKey[r]);

        left = tmp;
    }
    for (int i = 0; i < 32; i++) res[i] = left[i];
    for (int i = 32; i < 64; i++) res[i] = right[i - 32];
    cur = res;
    for (int i = 0; i < 64; i++) res[63 - i] = cur[64 - IP_1[i]];
    return res;
}

BITS64 decrypt(BITS64 ori) {
    BITS64 res, cur;
    BITS32 left, right;
    for (int i = 0; i < 64; i++) cur[63 - i] = ori[64 - IP[i]];
    for (int i = 32; i < 64; i++) left[i - 32] = cur[i];
    for (int i = 0; i < 32; i++) right[i] = cur[i];
    for (int r = 0; r < 16; r++) {
        BITS32 tmp = right;
        right = left ^ f(right, subKey[16 - r - 1]);
        left = tmp;
    }
    for (int i = 0; i < 32; i++) res[i] = left[i];
    for (int i = 32; i < 64; i++) res[i] = right[i - 32];
    cur = res;
    for (int i = 0; i < 64; i++) res[63 - i] = cur[64 - IP_1[i]];
    return res;
}
```

- **Encode()**和 **Decode()**主要完成对源文本的分组工作，同时，在 01 串长度不为 64 的整数倍时，也完成了将之补齐并记录补齐位数的工作。同时，两组函数也实现了 CBC 分组加密的功能。

```cpp
    void Encode() {
        int k = len % 64, st = len - k, res = 64 - k;
        if (res != 64) {
            if (res >= 6) {
                len += res;
                res -= 6;
                for (int i = 0; i < 6; i++)
                    text[st + 58 + i] = res & 1, res >>= 1;
            } else {
                len += res + 64;
                res -= 6;
                for (int i = 0; i < 6; i++)
                    text[st + 122 + i] = res & 1, res >>= 1;
            }
        } else {
            len += 64;
            res -= 6;
            for (int i = 0; i < 6; i++)
                text[st + 58 + i] = res & 1, res >>= 1;
        }
# ifdef DEBUG
        for (int i = 0; i < len; i++) printf("%d", text[i]);
        putchar('\n');
# endif
        generateKeys();
        BITS64 s, t;
        for (int i = 0; i < len; i += 64) {
            for (int j = 0; j < 64; j++) s[j] = text[i + j]^pre[j];
            t = encrypt(s);
            for (int j = 0; j < 64; j++) pre[j] = to[i + j] = t[j];
        }
    }

    void Decode() {
        generateKeys();
        BITS64 s, t;
        bool tmp[64];
        for (int i = 0; i < len; i += 64) {
            for (int j = 0; j < 64; j++) s[j] = text[i + j];
            t = decrypt(s);
            memcpy(tmp, text + i, 64);
            for (int j = 0; j < 64; j++) to[i + j] = t[j]^pre[j];
            memcpy(pre, tmp, 64);
        }
        int l = 0;
        for (int i = 0; i < 6; i++)
            l = l * 2 + to[len - i - 1];
# ifdef DEBUG
        for (int i = 0; i < len; i++) printf("%d", to[i]);
        putchar('\n');
        printf("%d\n", len);
# endif
        len -= l + 6;
    }
```

- 主函数 **main()**中主要完成用户交互和输入检查的工作

```cpp
int main() {
    while (true) {
        while (true) {
            system("cls");
            printf("Please Select:\n1. Encode\n2. Decode\n0. Exit\n");
            scanf("%d", &kase);
            if (kase == 0) {
                system("cls");
                printf("Successfully exit!\n");
                Sleep(2000);
                return 0;
            }
            if (0 <= kase && kase <= 2) break;
            printf("Please enter again!\n");
            Sleep(2000);
        }
        while (true) {
            system("cls");
            printf("%s\n", kase == 1? " - - - Encoding - - - ": " - - - Decoding - - - ");
            printf("Please enter the key (64 bits): ");
            scanf("%s", s);
            bool check = true;
            int n = strlen(s);
            if (n != 64) check = false;
            for (int i = 0; i < 64 && check; i++)
                if (s[i] != '0' && s[i] != '1') check = false;
            if (check) {
                for (int i = 0; i < 64; i++) DES::key[i] = s[i] == '1'? 1: 0;
                break;
            }
            printf("Please enter valid key!\n");
            Sleep(2000);
        }
        while (true) {
            system("cls");
            printf("%s\n", kase == 1? " - - - Encoding - - - ": " - - - Decoding - - - ");
            printf("Please enter the file name you want to %scode: ", kase == 1? "en": "de");
            scanf("%s", s);
            if (!(f1 = fopen(s, "r"))) {
                printf("\nThere is no such file under current directory! Please enter again.\n");
                Sleep(2000);
                continue;
            }
            char c;
            bool check = true;
            for (DES::len = 0; ~fscanf(f1, "%c", &c); DES::len++) {
                if (DES::len >= 1000 || (c != '0' && c != '1')) {
                    check = false;
                    break;
                }
                DES::text[DES::len] = c == '1'? 1: 0;
            }
            if (!check) {
                printf("\nNote: There can be less than 1000 digits of 0/1 in the input file. Please check!\n");
                Sleep(4000);
                continue;
            }
            break;
        }
        fclose(f1);
        while (true) {
            system("cls");
            printf("%s\n", kase == 1? " - - -  Encoding - -  - ": " - - -  Decoding - -  - ");
            printf("Please enter the file name you want to save %scoded text: ", kase == 1 ? "en": "de");
            scanf("%s", s);
            f2 = fopen(s, "w");
            break;
        }
        if (kase == 1) DES::Encode();
        else DES::Decode();
        DES::out(f2);
        fclose(f2);
        printf("Succeed!\n");
        Sleep(2000);
    }
    return 0;
}
```

Form1.cs

以下是 GUI 实现代码

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;
using System.Windows.Forms;


namespace DES
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        public string input = "", output = "";

        private bool Check()
        {
            string Key = tbKey.Text;
            string IV = tbIV.Text;
            bool res = true;
            Note0.Text = "";
            Note1.Text = "";
            Note2.Text = "";
            if (input == "")
            {
                Note0.Text = "Please Select File!";
                res = false;
            }
            if (Key.Length != 64)
            {
                Note1.Text = "Only 64bits 0/1 string acceptable!";
                res = false;
            }
            for (int i = 0; i < Key.Length; i++)
                if (Key[i] != '0' && Key[i] != '1')
                {
                    Note1.Text = "Only 64bits 0/1 string acceptable!";
                    res = false;
                    break;
                }
            if (IV.Length != 64)
            {
                Note2.Text = "Only 64bits 0/1 string acceptable!";
                res = false;
            }
            for (int i = 0; i < IV.Length; i++)
                if (IV[i] != '0' && IV[i] != '1')
                {
                    Note2.Text = "Only 64bits 0/1 string acceptable!";
                    res = false;
                    break;
                }
            return res;
        }

        private void FileBotton_Click(object sender, EventArgs e)
        {
            OpenFileDialog fileDialog = new OpenFileDialog();
            fileDialog.Multiselect = false;
            fileDialog.Title = "Please Select File";
            fileDialog.Filter = "Text File(*.txt)|*.txt";
            fileDialog.InitialDirectory = Application.StartupPath;
            if (fileDialog.ShowDialog() == DialogResult.OK)
            {
                string display = fileDialog.FileName;
                input = Filename.Text = display;
            }
        }
        private void encodeBotton_Click(object sender, EventArgs e)
        {
            string Key = tbKey.Text;
            string IV = tbIV.Text;
            if (!Check()) return;

            SaveFileDialog SaveData = new SaveFileDialog();
            SaveData.Title = "Select File";
            SaveData.InitialDirectory = Application.StartupPath;
            SaveData.Filter = "Text File(*.txt)|*.txt";
            SaveData.FileName = "cipher";
            if (SaveData.ShowDialog() == DialogResult.OK)
            {
                output = SaveData.FileName;
            }
            else
            {
                return;
            }

            Process p = new Process();
            p.StartInfo.CreateNoWindow = true;          // 不创建新窗口
            p.StartInfo.UseShellExecute = false;        // 不启用 shell 启动进程
            p.StartInfo.RedirectStandardInput = true;   // 重定向输入
            p.StartInfo.RedirectStandardOutput = true; // 重定向标准输出
            p.StartInfo.RedirectStandardError = true;   // 重定向错误输出
            p.StartInfo.FileName = "desCore.exe";
            p.Start();
            p.StandardInput.WriteLine("1");
            p.StandardInput.WriteLine(Key);
            p.StandardInput.WriteLine(IV);
            p.StandardInput.WriteLine(input);
            p.StandardInput.WriteLine(output);
            p.StandardInput.WriteLine("0");
            MessageBox.Show("Successful!");
            p.Close();
        }

        private void decodeBotton_Click(object sender, EventArgs e)
        {
            string Key = tbKey.Text;
            string IV = tbIV.Text;
            if (!Check()) return;

            SaveFileDialog SaveData = new SaveFileDialog();
            SaveData.Title = "Select File";
            SaveData.InitialDirectory = Application.StartupPath;
            SaveData.Filter = "Text File(*.txt)|*.txt";
            SaveData.FileName = "plain";
            if (SaveData.ShowDialog() == DialogResult.OK)
            {
                output = SaveData.FileName;
            }
            else
            {
                return;
            }

            Process p = new Process();
            p.StartInfo.CreateNoWindow = true;          // 不创建新窗口
            p.StartInfo.UseShellExecute = false;        // 不启用 shell 启动进程
            p.StartInfo.RedirectStandardInput = true;   // 重定向输入
            p.StartInfo.RedirectStandardOutput = true; // 重定向标准输出
            p.StartInfo.RedirectStandardError = true;   // 重定向错误输出
            p.StartInfo.FileName = "desCore.exe";
            p.Start();
            p.StandardInput.WriteLine("2");
            p.StandardInput.WriteLine(Key);
            p.StandardInput.WriteLine(IV);
            p.StandardInput.WriteLine(input);
            p.StandardInput.WriteLine(output);
            p.StandardInput.WriteLine("0");
            MessageBox.Show("Successful!");
            p.Close();
        }

        private void Show1_Click(object sender, EventArgs e)
        {
            if (tbKey.PasswordChar == '*')
            {
                tbKey.PasswordChar = '\0';
                Show1.Text = "hide";
            }
            else
            {
                tbKey.PasswordChar = '*';
                Show1.Text = "show";
            }
        }

        private void Show2_Click(object sender, EventArgs e)
        {
            if (tbIV.PasswordChar == '*')
            {
                tbIV.PasswordChar = '\0';
                Show2.Text = "hide";
            }
            else
            {
                tbIV.PasswordChar = '*';
                Show2.Text = "show";
            }
        }
    }
}
```