杭州电子科技大学

实验报告

课程名称：密码学课程设计　　姓名：苏展　　　学号：　18271329

实验地点：科技馆 620　　　　　　　　　实验时间：2020-4-16

一、实验名称：SM4 密码实验

二、实验要求：

1、熟悉分组密码的基本框架。

2、掌握 SM4 密码的加解密原理。

3、用 Visual C++实现 SM4 密码程序并输出结果。

三、实验内容：

SM4 算法是我国商用密码标准，其前身是 SMS4 算法。SM4 算法是一个分组加密算法，分组长度和密钥长度均 128bit。SM4 算法使用 32 轮的非线性迭代结构。SM4 在最后一轮非线性迭代之后加上了一个反序变换，因此 SM4 中只要解密密钥是加密密钥的逆序，它的解密算法与加密算法就可以保持一致。SM4 的主体运算是非平衡 Feistel 网络。整体逻辑结构如图 1 所示，经过 32 轮变换把明文变换为密文。

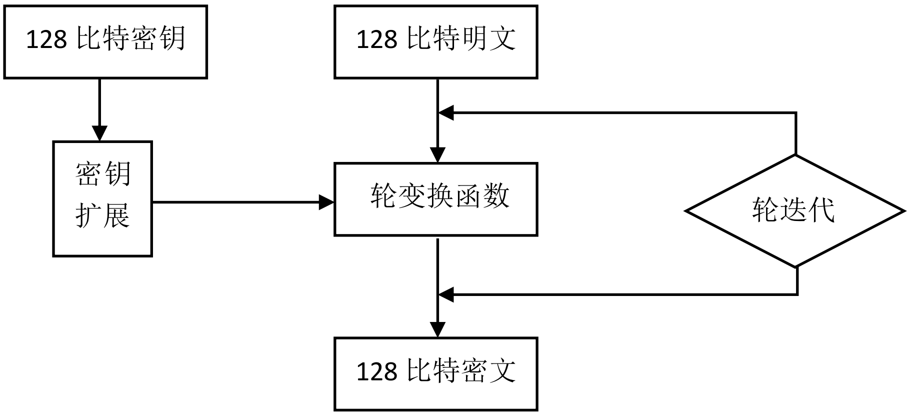


图 1　SM4 算法总体流程

其中密钥扩展运算把 128 bit 的种子密钥扩展为 32 个 32 bit 的子密钥。下面分别介绍轮函数、密钥扩展和加解密。

1. 轮函数

轮函数的规则由 $X_{i+4}=X_i\oplus T(X_{i+1}\oplus X_{i+2}\oplus X_{i+3}\oplus RK_i)$ 给出，其中 i = 0,1,…31。第 i 轮的输入为 $(X_i,X_{i+1},X_{i+2},X_{i+3})$，输出为 $(X_{i+1},X_{i+2},X_{i+3},X_{i+4})$。第一轮的输入 (X_0,X_1,X_2,X_3)，即为 128 bit 明文的四个分组，最后一轮的输出 $(X_{32},X_{33},X_{34},X_{35})$，再经过逆序，得到了密文 $(X_{35},X_{34},X_{33},X_{32})$。下图即为轮函数的结构。

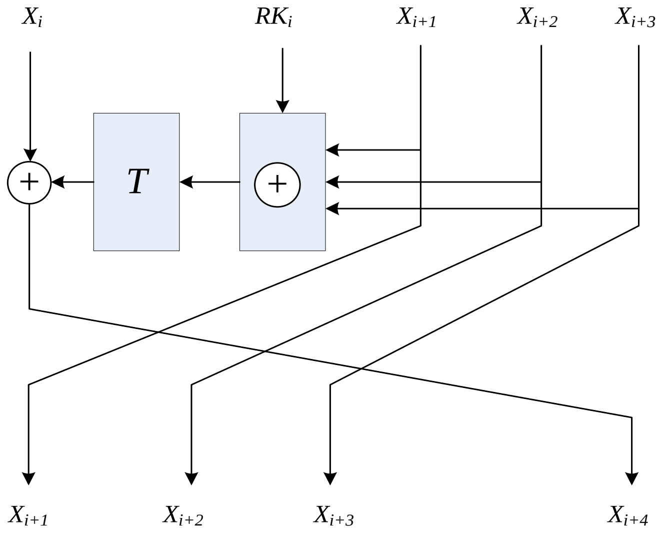


图 2　SM4 算法轮函数

SM4 算法的合成置换 *T* 是 $F_2^{32}\to F_2^{32}$ 的可逆置换。*T* 置换是由一个非线性变换 τ 和一个线性扩散变换 *L* 复合而成，即 $T(\cdot)=L(\tau(\cdot))$。*T* 置换的过程如下图所示：

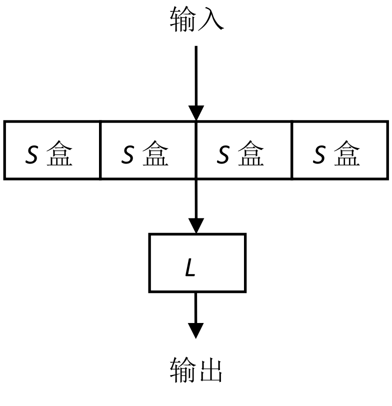


图 3　SM4 算法 *L* 函数

非线性变换 τ 由四个 *S* 盒并行组成。设变换 τ 的输入 $A=(a_0,a_1,a_2,a_3)\in(F_2^8)^4$，

输出是$B=(b_0,b_1,b_2,b_3)\in(F_2^8)^4$，则$(b_0,b_1,b_2,b_3)=(S(a_0),S(a_1),S(a_2),S(a_3))$。不同于DES 等分组密码算法，SM4 算法中的这四个 *S* 盒实际上是同一个 8 bit -> 8bit 的 *S* 盒，详见下表：

表 1　SM4 算法的 S 盒

	0*x*0	0*x*1	0*x*2	0*x*3	0*x*4	0*x*5	0*x*6	0*x7*	0*x8*	0*x9*	0a*x*	0*xb*	0*xc*	0*xd*	0e*x*	0*xf*
0*x*0	D6	90	E9	FE	CC	E1	3D	B7	16	B6	14	C2	28	FB	2C	05
0*x*1	2B	67	9A	76	2A	BE	04	C3	AA	44	13	26	49	86	06	99
0*x*2	9C	42	50	F4	91	EF	98	7A	33	54	0B	43	ED	CF	AC	62
0*x*3	E4	B3	1C	A9	C9	08	E8	95	80	DF	94	FA	75	8F	3F	A6
0*x*4	47	07	A7	FC	F3	73	17	BA	83	59	3C	19	E6	85	4F	A8
0*x*5	68	6B	81	B2	71	64	DA	8B	F8	EB	0F	4B	70	56	9D	35
0*x*6	1E	24	0E	5E	63	58	D1	A2	25	22	7C	3B	01	21	78	87
0*x*7	D4	00	46	57	9F	D3	27	52	4C	36	02	E7	A0	C4	C8	9E
0*x*8	EA	BF	8A	D2	40	C7	38	B5	A3	F7	F2	CE	F9	61	15	A1
0*x*9	E0	AE	5D	A4	9B	34	1A	55	AD	93	32	30	F5	8C	B1	E3
0*xa*	1D	F6	E2	2E	82	66	CA	60	C0	29	23	AB	0D	53	4E	6F
0*xb*	D5	DB	37	45	DE	FD	8E	2F	03	FF	6A	72	6D	6C	5B	51
0*xc*	8D	1B	AF	92	BB	DD	BC	7F	11	D9	5C	41	1F	10	5A	D8
0*xd*	0A	C1	31	88	A5	CD	7B	BD	2D	74	D0	12	B8	E5	B4	B0
0*xe*	89	69	97	4A	0C	96	77	7E	65	B9	F1	09	C5	6E	C6	84
0*xf*	18	F0	7D	EC	3A	DC	4D	20	79	EE	5F	3E	D7	CB	39	48

其中左边的列表示 8 bit 输入的高位部分，上方的行表示 8 bit 输入的低位部分。

非线性变换τ的输出是线性变换 *L* 的输入，设 *L* 的输入为$B\in F_2^{32}$，输出为$C\in F_2^{32}$，则

$$C=L(B)=B\oplus(B<<<2)\oplus(B<<<10)\oplus(B<<<18)\oplus(B<<<24).$$

其中 B<<<n 表示 B 循环左移 n 位。

2. 密钥扩展

在密钥扩展方案中，种子密钥经过扩展生成 32 个轮密钥，每个轮密钥长度为 32 bit。首先，128 bit 的种子密钥 *SK* 分为四组 $SK=(SK_0,SK_1,SK_2,SK_3)\in(F_2^8)^4$，再给定系统参数

$$FK=(FK_0,FK_1,FK_2,FK_3)=(0xa3b1bac6,\ 0x56aa3350,\ 0x677d9197,\ 0xb270022dc)$$

与固定参数 $CK_i=(ck_{i0},ck_{i1},ck_{i2},ck_{i3})\in(F_2^8)^4$（其中 $ck_{ij}=7(4i+j)\bmod 256$），密钥扩展规则如下：

$$\begin{aligned}&(K_0,K_1,K_2,K_3)=(SK_0\oplus FK_0,SK_1\oplus FK_1,SK_2\oplus FK_2,SK_3\oplus FK_3)\\&RK_i=K_{i+4}=K_i\oplus T'(K_{i+1}\oplus K_{i+2}\oplus K_{i+3}\oplus CK_i)\end{aligned}$$

其中 i = 0,1,2,…31 用于生成共 32 个轮密钥。*T'*变换与加密算法轮函数中的变换除线性变换 *L* 不同外，其他相同。*T'*变换中的线性变换 *L'*为

$$L'(B)=B\oplus(B<<<13)\oplus(B<<<23).$$

3. 加解密

加密与解密的轮函数结构完全相同，唯一的区别是解密密钥是加密密钥的逆序。加密轮密钥的使用顺序为 $(RK_0,\cdots,RK_{31})$，解密时轮密钥的使用顺序为 $(RK_{31},\cdots,RK_0)$。

4. 主要步骤

1) 新建一个空项目，取名 sm4_test。

2) 在左边的解决方案资源管理器中添加 cpp 文件，取名为 sm4_test.cpp。

3) 在 SM4 中，32bit 可用一个无符号整型(unsigned int)变量表示，128bit 可用长度为 4 的无符号整型数组表示。

4) 在 sm4_test.cpp 中先写入

```
    #include<iostream>
    using namespace std;
    static unsigned int RK[32] = {0};        //32 轮轮密钥，每个 RKi 为 32bit
    static unsigned int K[36] = {0};    //在密钥扩展中使用，一共 36 组，每个 Ki 为 32bit
    static const unsigned char S_Box[256] = {       //S 盒，输入 8bit，输出 8bit
0xd6,0x90,0xe9,0xfe,0xcc,0xe1,0x3d,0xb7,0x16,0xb6,0x14,0xc2,0x28,0xfb,0x2c,0x05,
0x2b,0x67,0x9a,0x76,0x2a,0xbe,0x04,0xc3,0xaa,0x44,0x13,0x26,0x49,0x86,0x06,0x99,
0x9c,0x42,0x50,0xf4,0x91,0xef,0x98,0x7a,0x33,0x54,0x0b,0x43,0xed,0xcf,0xac,0x62,
0xe4,0xb3,0x1c,0xa9,0xc9,0x08,0xe8,0x95,0x80,0xdf,0x94,0xfa,0x75,0x8f,0x3f,0xa6,
0x47,0x07,0xa7,0xfc,0xf3,0x73,0x17,0xba,0x83,0x59,0x3c,0x19,0xe6,0x85,0x4f,0xa8,
0x68,0x6b,0x81,0xb2,0x71,0x64,0xda,0x8b,0xf8,0xeb,0x0f,0x4b,0x70,0x56,0x9d,0x35,
0x1e,0x24,0x0e,0x5e,0x63,0x58,0xd1,0xa2,0x25,0x22,0x7c,0x3b,0x01,0x21,0x78,0x87,
0xd4,0x00,0x46,0x57,0x9f,0xd3,0x27,0x52,0x4c,0x36,0x02,0xe7,0xa0,0xc4,0xc8,0x9e,
0xea,0xbf,0x8a,0xd2,0x40,0xc7,0x38,0xb5,0xa3,0xf7,0xf2,0xce,0xf9,0x61,0x15,0xa1,
0xe0,0xae,0x5d,0xa4,0x9b,0x34,0x1a,0x55,0xad,0x93,0x32,0x30,0xf5,0x8c,0xb1,0xe3,
0x1d,0xf6,0xe2,0x2e,0x82,0x66,0xca,0x60,0xc0,0x29,0x23,0xab,0x0d,0x53,0x4e,0x6f,
0xd5,0xdb,0x37,0x45,0xde,0xfd,0x8e,0x2f,0x03,0xff,0x6a,0x72,0x6d,0x6c,0x5b,0x51,
0x8d,0x1b,0xaf,0x92,0xbb,0xdd,0xbc,0x7f,0x11,0xd9,0x5c,0x41,0x1f,0x10,0x5a,0xd8,
0x0a,0xc1,0x31,0x88,0xa5,0xcd,0x7b,0xbd,0x2d,0x74,0xd0,0x12,0xb8,0xe5,0xb4,0xb0,
0x89,0x69,0x97,0x4a,0x0c,0x96,0x77,0x7e,0x65,0xb9,0xf1,0x09,0xc5,0x6e,0xc6,0x84,
0x18,0xf0,0x7d,0xec,0x3a,0xdc,0x4d,0x20,0x79,0xee,0x5f,0x3e,0xd7,0xcb,0x39,0x48};
    static unsigned int CK[32] = {0};        //定义固定参数 CK, 32 组，每个 CKi 为 32bit
    static unsigned int FK[4] = { 0xa3b1bac6, 0x56aa3350, 0x677d9197, 0xb27022dc};
                                             //给定的系统参数 FK, 4 组，每个 FKi 为 32bit
```

4) 编写各个模块函数，例如：

```
    unsigned int S_Func(unsigned int In);                       //S 函数
    void SetRoundKey(unsigned int SK[]);                        //密钥扩展函数
    unsigned int RotL(unsigned int In, int loop);               //循环左移函数
    unsigned int T(unsigned int In);                            //T 置换
    unsigned int T1(unsigned int In);                           //T'置换
    unsigned int L_Func(unsigned int In);                       //L 变换
    unsigned int L1_Func(unsigned int In);                      //L'变换
    void Crypt(unsigned int Out[], unsigned int In[], unsigned int K, bool flag);   //加解密
    void SetPara();                                             //设置固定参数 CK
```

5) 编写主函数，调试程序，目的是测试各个函数编写正确。SM4 的标准文

档里给出一下的测试示例：

明文　=　0x0123456789abcdeffedcba9876543210 (128bit)

密钥　=　0x0123456789abcdeffedcba9876543210 (128bit)

则密文 =　0x681edf34d206965e86b3e94f536e4246 (128bit)

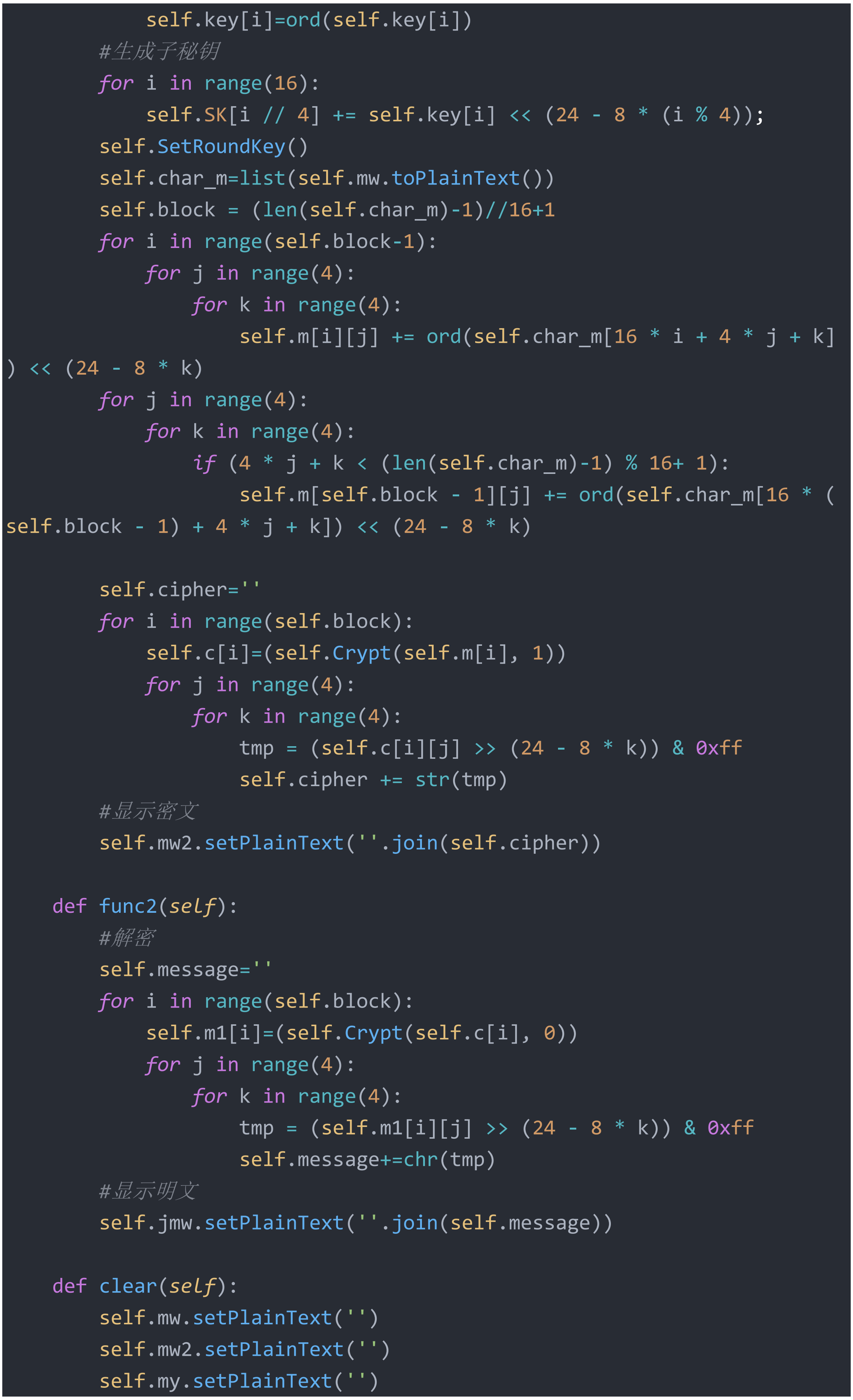
6) 所有函数测试都正确后，新建一个对话框项目 SM4，包含明文框、密钥框、密文框、解密文框等，以及加密，解密，清空按钮，将以上的全局变量、S 盒以及所有子函数声明及实现代码全部复制到 SM4Dlg.cpp 中的合适位置，再编写按钮事件，实现加解密功能。

部分函数参考代码：

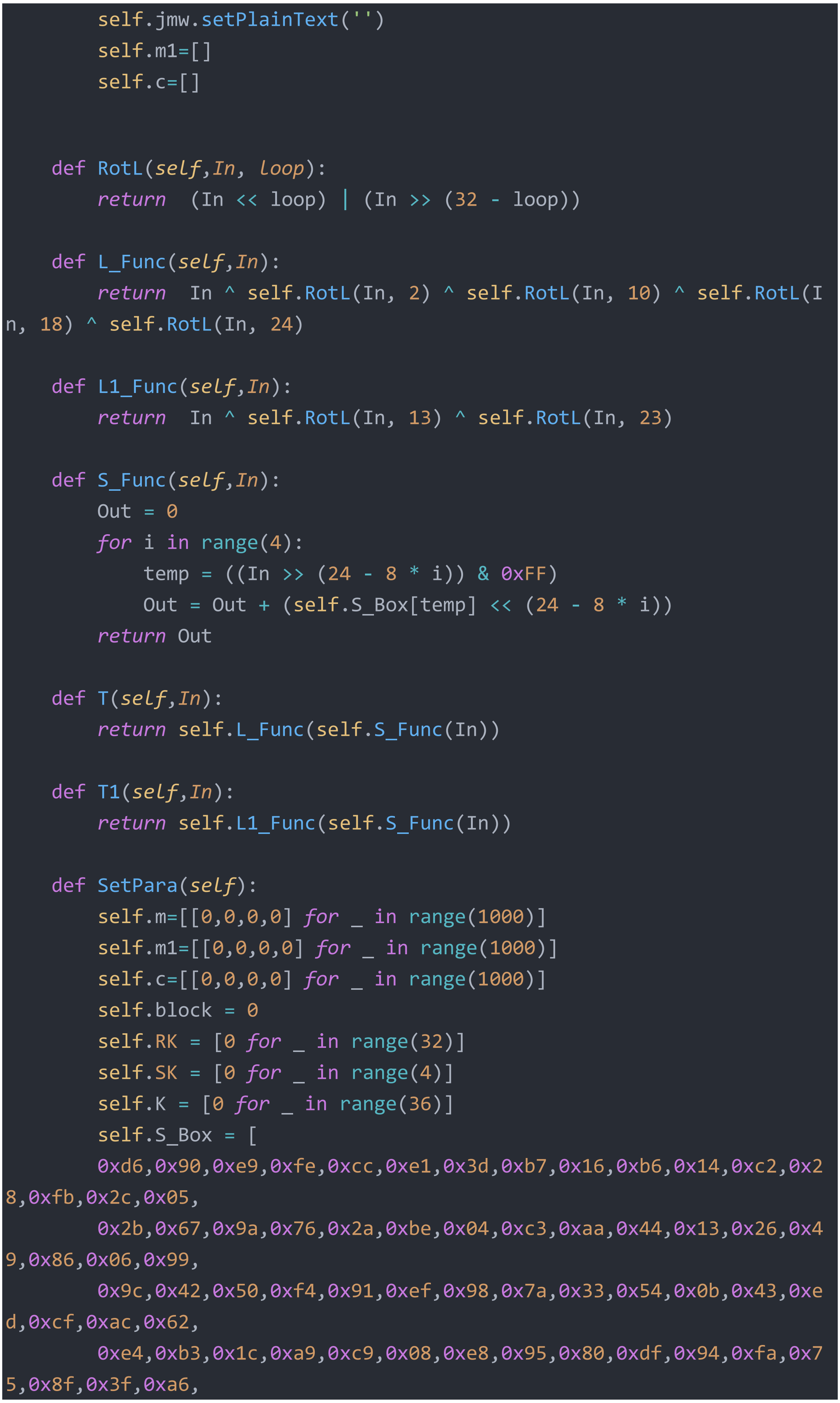
```
unsigned int RotL(unsigned int In, int loop)
{
    return    (In << loop) | (In >> (32 - loop));
}

unsigned int L_Func(unsigned int In)
{
    return   In ^ RotL(In,2) ^ RotL(In,10) ^ RotL(In,18) ^ RotL(In,24);
}

unsigned int L1_Func(unsigned int In)
{
    return   In ^ RotL(In,13) ^ RotL(In,23);
}

unsigned int S_Func(unsigned int In)
{
    unsigned int Out = 0;
    unsigned char temp = {0};
    for(int i = 0; i<4; i++)                    //4 组 8bit 的部分分别经过 S 盒，再拼成 32bit
    {
        temp = ((In >> (24 - 8 * i)) & 0xFF);
        Out = Out + (S_Box[temp] << (24 - 8 * i));
    }
    return Out;
}

unsigned int T(unsigned int In)
{
        return L_Func(S_Func(In));
}

unsigned int T1(unsigned int In)
{
        return L1_Func(S_Func(In));
}

void SetPara()                                    //设置所有固定参数 CKi
{
        unsigned int temp = 0;
        for(int i = 0; i<32; i++)
        {
                for(int j = 0; j <4; j++)
                {
                        temp = (7 * (4 * i + j)) & 0xFF;
                        temp = temp << (24 - 8 * j);
                        CK[i] = CK[i] + temp;
                }
        }
}
```

SM4 密码程序代码如下：(所有模块函数代码以及按钮事件代码)

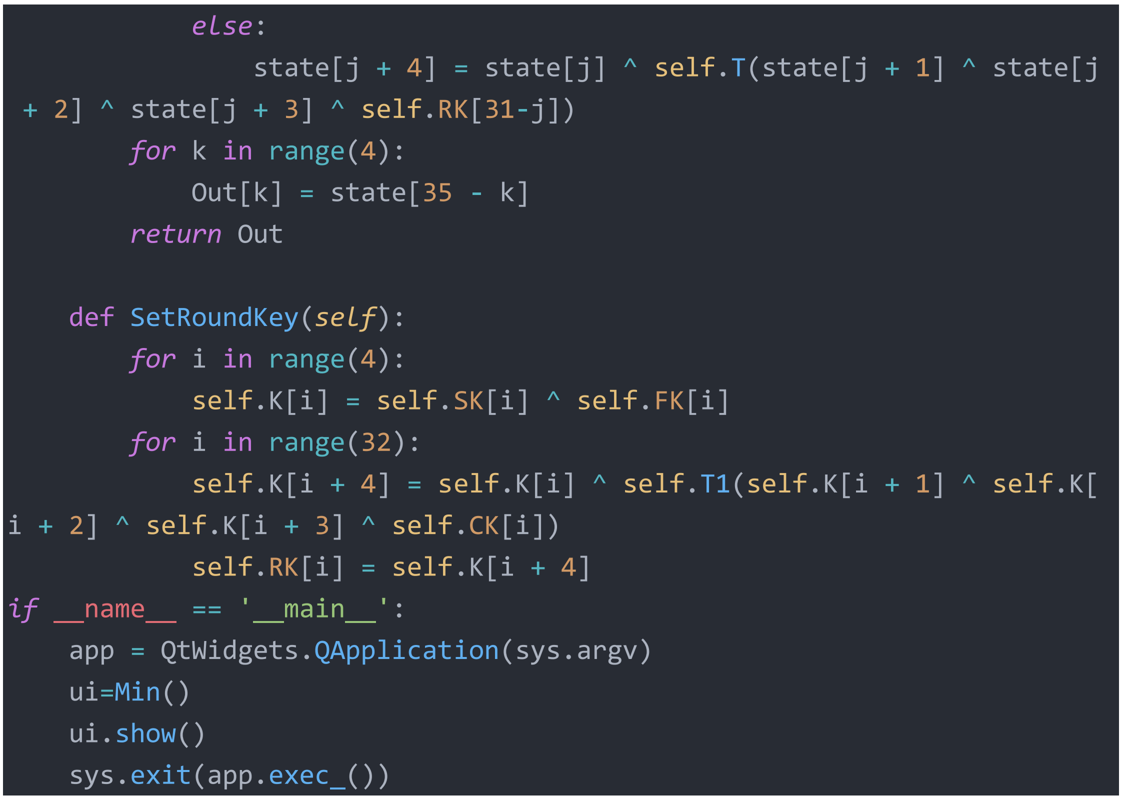
```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

from PyQt5 import QtCore, QtGui, QtWidgets
import sys
from Ui_SM4 import Ui_MainWindow

class Min(QtWidgets.QMainWindow,Ui_MainWindow):
    def __init__(self,parent=None): #ui 部分
        super().__init__()
        self.setupUi(self)
        self.jm.clicked.connect(self.func1)
        self.jm2.clicked.connect(self.func2)
        self.qk.clicked.connect(self.clear)

    def func1(self):
        #设置参数
        self.SetPara()
        #读取密钥
        self.key=list(self.my.toPlainText())
        for i in range(len(self.key)):
```



```
            self.key[i]=ord(self.key[i])
        #生成子秘钥
        for i in range(16):
            self.SK[i // 4] += self.key[i] << (24 - 8 * (i % 4));
        self.SetRoundKey()
        self.char_m=list(self.mw.toPlainText())
        self.block = (len(self.char_m)-1)//16+1
        for i in range(self.block-1):
            for j in range(4):
                for k in range(4):
                    self.m[i][j] += ord(self.char_m[16 * i + 4 * j + k]
) << (24 - 8 * k)
        for j in range(4):
            for k in range(4):
                if (4 * j + k < (len(self.char_m)-1) % 16+ 1):
                    self.m[self.block - 1][j] += ord(self.char_m[16 * (
self.block - 1) + 4 * j + k]) << (24 - 8 * k)

        self.cipher=''
        for i in range(self.block):
            self.c[i]=(self.Crypt(self.m[i], 1))
            for j in range(4):
                for k in range(4):
                    tmp = (self.c[i][j] >> (24 - 8 * k)) & 0xff
                    self.cipher += str(tmp)
        #显示密文
        self.mw2.setPlainText(''.join(self.cipher))

    def func2(self):
        #解密
        self.message=''
        for i in range(self.block):
            self.m1[i]=(self.Crypt(self.c[i], 0))
            for j in range(4):
                for k in range(4):
                    tmp = (self.m1[i][j] >> (24 - 8 * k)) & 0xff
                    self.message+=chr(tmp)
        #显示明文
        self.jmw.setPlainText(''.join(self.message))

    def clear(self):
        self.mw.setPlainText('')
        self.mw2.setPlainText('')
        self.my.setPlainText('')
```

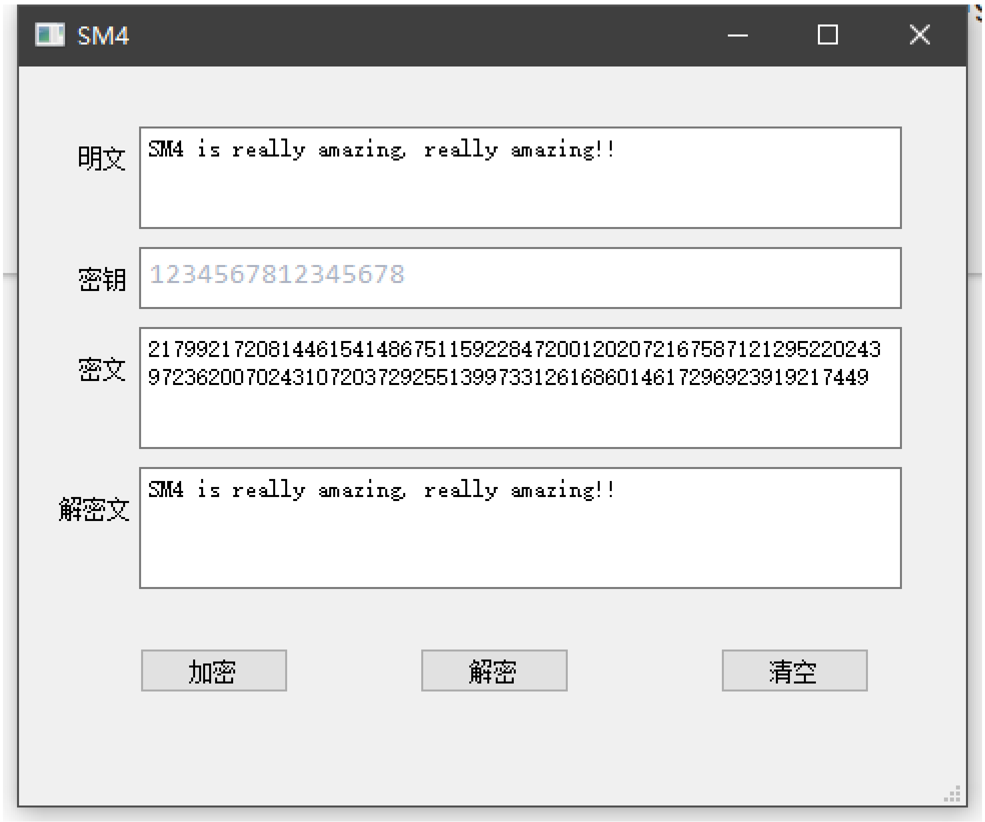


```
        self.jmw.setPlainText('')
        self.m1=[]
        self.c=[]


    def RotL(self,In, loop):
        return  (In << loop) | (In >> (32 - loop))

    def L_Func(self,In):
        return  In ^ self.RotL(In, 2) ^ self.RotL(In, 10) ^ self.RotL(In, 18) ^ self.RotL(In, 24)

    def L1_Func(self,In):
        return  In ^ self.RotL(In, 13) ^ self.RotL(In, 23)

    def S_Func(self,In):
        Out = 0
        for i in range(4):
            temp = ((In >> (24 - 8 * i)) & 0xFF)
            Out = Out + (self.S_Box[temp] << (24 - 8 * i))
        return Out

    def T(self,In):
        return self.L_Func(self.S_Func(In))

    def T1(self,In):
        return self.L1_Func(self.S_Func(In))

    def SetPara(self):
        self.m=[[0,0,0,0] for _ in range(1000)]
        self.m1=[[0,0,0,0] for _ in range(1000)]
        self.c=[[0,0,0,0] for _ in range(1000)]
        self.block = 0
        self.RK = [0 for _ in range(32)]
        self.SK = [0 for _ in range(4)]
        self.K = [0 for _ in range(36)]
        self.S_Box = [
        0xd6,0x90,0xe9,0xfe,0xcc,0xe1,0x3d,0xb7,0x16,0xb6,0x14,0xc2,0x28,0xfb,0x2c,0x05,
        0x2b,0x67,0x9a,0x76,0x2a,0xbe,0x04,0xc3,0xaa,0x44,0x13,0x26,0x49,0x86,0x06,0x99,
        0x9c,0x42,0x50,0xf4,0x91,0xef,0x98,0x7a,0x33,0x54,0x0b,0x43,0xed,0xcf,0xac,0x62,
        0xe4,0xb3,0x1c,0xa9,0xc9,0x08,0xe8,0x95,0x80,0xdf,0x94,0xfa,0x75,0x8f,0x3f,0xa6,
        0x47,0x07,0xa7,0xfc,0xf3,0x73,0x17,0xba,0x83,0x59,0x3c,0x19,0xe6,0x85,0x4f,0xa8,
        0x68,0x6b,0x81,0xb2,0x71,0x64,0xda,0x8b,0xf8,0xeb,0x0f,0x4b,0x70,0x56,0x9d,0x35,
        0x1e,0x24,0x0e,0x5e,0x63,0x58,0xd1,0xa2,0x25,0x22,0x7c,0x3b,0x01,0x21,0x78,0x87,
        0xd4,0x00,0x46,0x57,0x9f,0xd3,0x27,0x52,0x4c,0x36,0x02,0xe7,0xa0,0xc4,0xc8,0x9e,
        0xea,0xbf,0x8a,0xd2,0x40,0xc7,0x38,0xb5,0xa3,0xf7,0xf2,0xce,0xf9,0x61,0x15,0xa1,
        0xe0,0xae,0x5d,0xa4,0x9b,0x34,0x1a,0x55,0xad,0x93,0x32,0x30,0xf5,0x8c,0xb1,0xe3,
        0x1d,0xf6,0xe2,0x2e,0x82,0x66,0xca,0x60,0xc0,0x29,0x23,0xab,0x0d,0x53,0x4e,0x6f,
        0xd5,0xdb,0x37,0x45,0xde,0xfd,0x8e,0x2f,0x03,0xff,0x6a,0x72,0x6d,0x6c,0x5b,0x51,
        0x8d,0x1b,0xaf,0x92,0xbb,0xdd,0xbc,0x7f,0x11,0xd9,0x5c,0x41,0x1f,0x10,0x5a,0xd8,
        0x0a,0xc1,0x31,0x88,0xa5,0xcd,0x7b,0xbd,0x2d,0x74,0xd0,0x12,0xb8,0xe5,0xb4,0xb0,
        0x89,0x69,0x97,0x4a,0x0c,0x96,0x77,0x7e,0x65,0xb9,0xf1,0x09,0xc5,0x6e,0xc6,0x84,
        0x18,0xf0,0x7d,0xec,0x3a,0xdc,0x4d,0x20,0x79,0xee,0x5f,0x3e,0xd7,0xcb,0x39,0x48 ]
        self.CK = [0 for _ in range(32)]
        self.FK = [ 0xa3b1bac6, 0x56aa3350, 0x677d9197, 0xb27022dc ]
        for i in range(32):
            for j in range(4):
                temp = (7 * (4 * i + j)) & 0xFF
                temp = temp << (24 - 8 * j)
                self.CK[i] = self.CK[i] + temp

    def Crypt(self, In,  flag):
        """
        加解密
        """
        Out=[0 for _ in range(4)]
        state = [0 for _ in range(36)]
        for i in range(4):
            state[i] = In[i]
        for j in range(32):
            if ( flag ==1 ):
                state[j + 4] = state[j] ^ self.T(state[j + 1] ^ state[j + 2] ^ state[j + 3] ^ self.RK[j])
```



```
            else:
                state[j + 4] = state[j] ^ self.T(state[j + 1] ^ state[j
 + 2] ^ state[j + 3] ^ self.RK[31-j])
        for k in range(4):
            Out[k] = state[35 - k]
        return Out

    def SetRoundKey(self):
        for i in range(4):
            self.K[i] = self.SK[i] ^ self.FK[i]
        for i in range(32):
            self.K[i + 4] = self.K[i] ^ self.T1(self.K[i + 1] ^ self.K[
i + 2] ^ self.K[i + 3] ^ self.CK[i])
            self.RK[i] = self.K[i + 4]
if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    ui=Min()
    ui.show()
    sys.exit(app.exec_())
```

SM4 密码输出界面截屏如下：



明文 SM4 is really amazing, really amazing!!

密钥 1234567812345678

密文 2179921720814461541486751159228472001202072167587121295220243
972362007024310720372925513997331261686014617296923919217449

解密文 SM4 is really amazing, really amazing!!