[3.1.SM4算法介绍 16](#_Toc13225)

[3.1.1 SM4中的术语说明 18](#_Toc1279)

[3.1.2 SM4中的数学原理 18](#_Toc1279)

**3.1 SM4算法介绍**

国密算法即国家密码局认定的国产密码算法，即商用密码。

商用密码，是指能够实现商用密码算法的加密、解密和认证等功能的技术。（包括密码算法编程技术和密码算法芯片、加密卡等的实现技术）。商用密码技术是商用密码的核心，国家将商用密码技术列入国家秘密，任何单位和个人都有责任和义务保护商用密码技术的秘密。

商用密码的应用领域十分广泛，主要用于对不涉及国家秘密内容但又具有敏感性的内部信息、行政事务信息、经济信息等进行加密保护。比如：商用密码可用于企业门禁管理、企业内部的各类敏感信息的传输加密、存储加密，防止非法第三方获取信息内容；也可用于各种安全认证、网上银行、数字签名等。

为了保障商用密码安全，国家商用密码管理办公室制定了一系列密码标准，包括SSF33、SM1（SCB2）、SM2、SM3、SM4、SM7、SM9、祖冲之密码算法等等。其中SSF33、SM1、SM4、SM7、祖冲之密码 是对称算法；SM2、SM9是非对称算法；SM3是哈希算法。

SM4算法作为国密算法的一种，是一个分组算法，用于无线局域网产品。该算法的分组长度为128比特，密钥长度为128比特。加密算法与密钥扩展算法都采用32轮非线性迭代结构。解密算法与加密算法的结构相同，只是轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。

**3.1.1 SM4算法的术语说明**

1.字与字节

用表示e比特的向量集，用中的元素表示字，中的元素称为字节。

2. S盒

S 盒为固定的8比特输入8比特输出的置换，记为*Sbox*(.)。

3. 基本运算

在本算法中采用了以下基本运算：

(1)⊕： 32 比特异或  
(2) <<<i：32 比特循环左移i位

4.密钥及密钥参量

加密密钥长度为 128 比特，表示为，其中为字。

轮密钥表示为其中为字。轮密钥由加密密钥生成。  
为系统参数，为固定参数，用于密钥扩  
展算法，其中、 为字。

**3.1.2 SM4算法的数学原理**

1.轮函数 F

本算法采用非线性迭代结构，以字为单位进行加密运算，称一次迭代运算为一轮变换。  
 设输入为，轮密钥为，则轮函数F为：

2合成置换

，是一个可逆变换，由非线性变换 τ 和线性变换 L 复合而成，即T(.)=L(τ(.))

1. 非线性变换τ

τ 由 4 个并行的 S 盒构成。  
 设输入为 ,输出为入为,则

1. 线性变换L

非线性变换 τ 的输出是线性变换L的输入。设输入为 ，输出为 ，则

2.2.2 S盒

S盒中数据均采用16进制表示。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| 0 | d6 | 90 | e9 | fe | cc | e1 | 3d | b7 | 16 | b6 | 14 | c2 | 28 | fb | 2c | 05 |
| 1 | 2b | 67 | 9a | 76 | 2a | be | 04 | c3 | aa | 44 | 13 | 26 | 49 | 86 | 06 | 99 |
| 2 | 9c | 42 | 50 | f4 | 91 | ef | 98 | 7a | 33 | 54 | 0b | 43 | ed | cf | ac | 62 |
| 3 | e4 | b3 | 1c | a9 | c9 | 08 | e8 | 95 | 80 | df | 94 | fa | 75 | 8f | 3f | a6 |
| 4 | 47 | 07 | a7 | fc | f3 | 73 | 17 | ba | 83 | 59 | 3c | 19 | e6 | 85 | 4f | a8 |
| 5 | 68 | 6b | 81 | b2 | 71 | 64 | da | 8b | f8 | eb | 0f | 4b | 70 | 56 | 9d | 35 |
| 6 | 1e | 24 | 0e | 5e | 63 | 58 | d1 | a2 | 25 | 22 | 7c | 3b | 01 | 21 | 78 | 87 |
| 7 | d4 | 00 | 46 | 57 | 9f | d3 | 27 | 52 | 4c | 36 | 02 | e7 | a0 | c4 | c8 | 9e |
| 8 | ea | bf | 8a | d2 | 40 | c7 | 38 | b5 | a3 | f7 | f2 | ce | f9 | 61 | 15 | a1 |
| 9 | e0 | ae | 5d | a4 | 9b | 34 | 1a | 55 | ad | 93 | 32 | 30 | f5 | 8c | b1 | e3 |
| a | 1d | f6 | e2 | 2e | 82 | 66 | ca | 60 | c0 | 29 | 23 | ab | 0d | 53 | 4e | 6f |
| b | d5 | db | 37 | 45 | de | fd | 8e | 2f | 03 | ff | 6a | 72 | 6d | 6c | 5b | 51 |
| c | 8d | 1b | af | 92 | bb | dd | bc | 7f | 11 | d9 | 5c | 41 | 1f | 10 | 5a | d8 |
| d | 0a | c1 | 31 | 88 | a5 | cd | 7b | bd | 2d | 74 | d0 | 12 | b8 | e5 | b4 | b0 |
| e | 89 | 69 | 97 | 4a | 0c | 96 | 77 | 7e | 65 | b9 | f1 | 09 | c5 | 6e | c6 | 84 |
| f | 18 | f0 | 7d | ec | 3a | dc | 4d | 20 | 79 | ee | 5f | 3e | d7 | cb | 39 | 48 |

例：输入‘ef’，则经S盒后的值为表中第e行和第f列的值，*Sbox*(‘ef’)= ‘84’。

2.3 加/解密算法

定义反序变换R为。

设明文输入为，密文输出为，轮密钥为。则本算法的加密变换为：

本算法的解密变换与加密变换结构相同，不同的仅是轮密钥的使用顺序。

加密时轮密钥的使用顺序为：（*rk*0, *rk*1, …, *rk*31）

解密时轮密钥的使用顺序为：（*rk*31, *rk*30, …, *rk*0）

2.4 密钥扩展算法

本算法中加密算法的轮密钥由加密密钥通过密钥扩展算法生成。

加密密钥MK=(MK0, MK1, MK2, MK3)，MK*i*∈，*i*＝0,1,2,3；

令*Ki*∈，*i*＝0,1,…,35，轮密钥为，则轮密钥生成方法为：

首先，(*K*0,*K*1,*K*2,*K*3)=(MK0⊕+FK0,MK1⊕+FK1,MK2⊕+FK2,MK3⊕+FK3) ，然后：

说明：

（1）'*T*变换与加密算法轮函数中的T基本相同，只将其中的线性变换L修改为以下： '*L:*

'*L*(B)=B○+(B<<<13)○+(B<<<23)；

（2）系统参数FK的取值，采用16进制表示为：

FK0=(A3B1BAC6)，FK1=(56AA3350)，FK2=(677D9197)，FK3=(B27022DC)

（3）固定参数CK的取值方法为：

设*cki,*j为*CKi*的第*j*字节（*i=*0,1,…,31；*j*=0,1,2,3），即*CKi* = (*cki*,0，*cki,*1，*cki*,2，*cki*,3) ∈，则*cki,j* = (*4i+j*)×7（mod 256）。32个固定参数*CKi*，其16进制表示为：

00070e15, 1c232a31, 383f464d, 545b6269,

70777e85, 8c939aa1, a8afb6bd, c4cbd2d9,

e0e7eef5, fc030a11, 181f262d, 343b4249,

50575e65, 6c737a81, 888f969d, a4abb2b9,

c0c7ced5, dce3eaf1, f8ff060d, 141b2229,

30373e45, 4c535a61, 686f767d, 848b9299,

a0a7aeb5, bcc3cad1, d8dfe6ed, f4fb0209,

10171e25, 2c333a41, 484f565d, 646b7279

2.5 实例

以下为本算法ECB工作方式的运算实例，用以验证密码算法实现的正确性。其中，数据采用16进制表示。

实例一：对一组明文用密钥加密一次

明 文: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10

加密密钥: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10

轮密钥与每轮输出状态：

rk[ 0] = f12186f9 X[ 0] = 27fad345

rk[ 1] = 41662b61 X[ 1] = a18b4cb2

rk[ 2] = 5a6ab19a X[ 2] = 11c1e22a

rk[ 3] = 7ba92077 X[ 3] = cc13e2ee

rk[ 4] = 367360f4 X[ 4] = f87c5bd5

rk[ 5] = 776a0c61 X[ 5] = 33220757

rk[ 6] = b6bb89b3 X[ 6] = 77f4c297

rk[ 7] = 24763151 X[ 7] = 7a96f2eb

rk[ 8] = a520307c X[ 8] = 27dac07f

rk[ 9] = b7584dbd X[ 9] = 42dd0f19

rk[10] = c30753ed X[10] = b8a5da02

rk[11] = 7ee55b57 X[11] = 907127fa

rk[12] = 6988608c X[12] = 8b952b83

rk[13] = 30d895b7 X[13] = d42b7c59

rk[14] = 44ba14af X[14] = 2ffc5831

rk[15] = 104495a1 X[15] = f69e6888

rk[16] = d120b428 X[16] = af2432c4

rk[17] = 73b55fa3 X[17] = ed1ec85e

rk[18] = cc874966 X[18] = 55a3ba22

rk[19] = 92244439 X[19] = 124b18aa

rk[20] = e89e641f X[20] = 6ae7725f

rk[21] = 98ca015a X[21] = f4cba1f9

rk[22] = c7159060 X[22] = 1dcdfa10

rk[23] = 99e1fd2e X[23] = 2ff60603

rk[24] = b79bd80c X[24] = eff24fdc

rk[25] = 1d2115b0 X[25] = 6fe46b75

rk[26] = 0e228aeb X[26] = 893450ad

rk[27] = f1780c81 X[27] = 7b938f4c

rk[28] = 428d3654 X[28] = 536e4246

rk[29] = 62293496 X[29] = 86b3e94f

rk[30] = 01cf72e5 X[30] = d206965e

rk[31] = 9124a012 X[31] = 681edf34

密文: 68 1e df 34 d2 06 96 5e 86 b3 e9 4f 53 6e 42 46

实例二：利用相同加密密钥对一组明文反复加密1000000次

明 文: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10

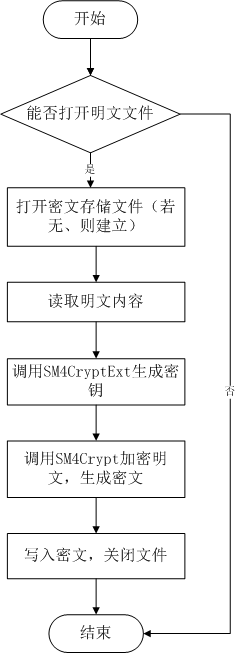
加密密钥: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10

密 文: 59 52 98 c7 c6 fd 27 1f 04 02 f8 04 c3 3d 3f 66

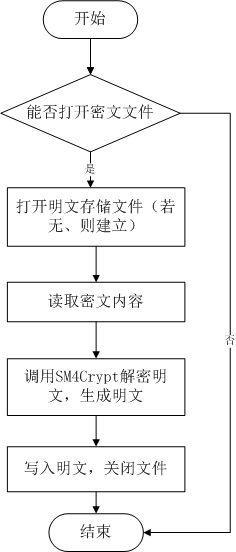
3.设计

程序流程：

加密流程：



解密流程：



4.实现

定义字和字节：

typedef unsigned char muint8;

typedef unsigned int muint32;

定义S盒：

const muint8 Sbox[256] = {

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

};

定义CK（密钥扩展算法中的参量）

const muint32 CK[32] = {

0x00070e15, 0x1c232a31, 0x383f464d, 0x545b6269,

0x70777e85, 0x8c939aa1, 0xa8afb6bd, 0xc4cbd2d9,

0xe0e7eef5, 0xfc030a11, 0x181f262d, 0x343b4249,

0x50575e65, 0x6c737a81, 0x888f969d, 0xa4abb2b9,

0xc0c7ced5, 0xdce3eaf1, 0xf8ff060d, 0x141b2229,

0x30373e45, 0x4c535a61, 0x686f767d, 0x848b9299,

0xa0a7aeb5, 0xbcc3cad1, 0xd8dfe6ed, 0xf4fb0209,

0x10171e25, 0x2c333a41, 0x484f565d, 0x646b7279 };

以下是在轮函数部分用到的数学变换

#define Rotl(\_x, \_y) (((\_x) << (\_y)) | ((\_x) >> (32 - (\_y)))) //循环移位

#define ByteSub(\_A) (Sbox[(\_A) >> 24 & 0xFF] << 24 ^ \

Sbox[(\_A) >> 16 & 0xFF] << 16 ^ \

Sbox[(\_A) >> 8 & 0xFF] << 8 ^ \

Sbox[(\_A) & 0xFF]) //L变换

#define L1(\_B) ((\_B) ^ Rotl(\_B, 2) ^ Rotl(\_B, 10) ^ Rotl(\_B, 18) ^ Rotl(\_B, 24))

#define L2(\_B) ((\_B) ^ Rotl(\_B, 13) ^ Rotl(\_B, 23))

SMS4的加解密函数

参数说明：Input为输入信息分组，Output为输出分组，rk为轮密钥

void SMS4Crypt(muint8 \*Input, muint8 \*Output, muint32 \*rk)

{

muint32 r, mid, x0, x1, x2, x3, \*p;

p = (muint32 \*)Input;

x0 = p[0];

x1 = p[1];

x2 = p[2];

x3 = p[3];

#ifdef LITTLE\_ENDIAN

x0 = Rotl(x0, 16); x0 = ((x0 & 0x00FF00FF) << 8) ^ ((x0 & 0xFF00FF00) >> 8);

x1 = Rotl(x1, 16); x1 = ((x1 & 0x00FF00FF) << 8) ^ ((x1 & 0xFF00FF00) >> 8);

x2 = Rotl(x2, 16); x2 = ((x2 & 0x00FF00FF) << 8) ^ ((x2 & 0xFF00FF00) >> 8);

x3 = Rotl(x3, 16); x3 = ((x3 & 0x00FF00FF) << 8) ^ ((x3 & 0xFF00FF00) >> 8);

#endif

for (r = 0; r < 32; r += 4)

{

mid = x1 ^ x2 ^ x3 ^ rk[r + 0];

mid = ByteSub(mid);

x0 ^= L1(mid);

mid = x2 ^ x3 ^ x0 ^ rk[r + 1];

mid = ByteSub(mid);

x1 ^= L1(mid);

mid = x3 ^ x0 ^ x1 ^ rk[r + 2];

mid = ByteSub(mid);

x2 ^= L1(mid);

mid = x0 ^ x1 ^ x2 ^ rk[r + 3];

mid = ByteSub(mid);

x3 ^= L1(mid);

}

#ifdef LITTLE\_ENDIAN

x0 = Rotl(x0, 16); x0 = ((x0 & 0x00FF00FF) << 8) ^ ((x0 & 0xFF00FF00) >> 8);

x1 = Rotl(x1, 16); x1 = ((x1 & 0x00FF00FF) << 8) ^ ((x1 & 0xFF00FF00) >> 8);

x2 = Rotl(x2, 16); x2 = ((x2 & 0x00FF00FF) << 8) ^ ((x2 & 0xFF00FF00) >> 8);

x3 = Rotl(x3, 16); x3 = ((x3 & 0x00FF00FF) << 8) ^ ((x3 & 0xFF00FF00) >> 8);

#endif

p = (muint32 \*)Output;

p[0] = x3;

p[1] = x2;

p[2] = x1;

p[3] = x0;

}

SMS4的密钥扩展算法

参数说明：Key为加密密钥，rk为子密钥，CryptFlag为加解密标志

void SMS4KeyExt(muint8 \*Key, muint32 \*rk, muint32 CryptFlag)

{

muint32 r, mid, x0, x1, x2, x3, \*p;

p = (muint32 \*)Key;

x0 = p[0];

x1 = p[1];

x2 = p[2];

x3 = p[3];

#ifdef LITTLE\_ENDIAN

x0 = Rotl(x0, 16); x0 = ((x0 & 0xFF00FF) << 8) ^ ((x0 & 0xFF00FF00) >> 8);

x1 = Rotl(x1, 16); x1 = ((x1 & 0xFF00FF) << 8) ^ ((x1 & 0xFF00FF00) >> 8);

x2 = Rotl(x2, 16); x2 = ((x2 & 0xFF00FF) << 8) ^ ((x2 & 0xFF00FF00) >> 8);

x3 = Rotl(x3, 16); x3 = ((x3 & 0xFF00FF) << 8) ^ ((x3 & 0xFF00FF00) >> 8);

#endif

x0 ^= 0xa3b1bac6;

x1 ^= 0x56aa3350;

x2 ^= 0x677d9197;

x3 ^= 0xb27022dc;

for (r = 0; r < 32; r += 4)

{

mid = x1 ^ x2 ^ x3 ^ CK[r + 0];

mid = ByteSub(mid);

rk[r + 0] = x0 ^= L2(mid);

mid = x2 ^ x3 ^ x0 ^ CK[r + 1];

mid = ByteSub(mid);

rk[r + 1] = x1 ^= L2(mid);

mid = x3 ^ x0 ^ x1 ^ CK[r + 2];

mid = ByteSub(mid);

rk[r + 2] = x2 ^= L2(mid);

mid = x0 ^ x1 ^ x2 ^ CK[r + 3];

mid = ByteSub(mid);

rk[r + 3] = x3 ^= L2(mid);

}

if (CryptFlag == DECRYPT)

{

for (r = 0; r < 16; r++)

mid = rk[r], rk[r] = rk[31 - r], rk[31 - r] = mid;

}

}

应用：

key为由声纹识别所得到的特征值。

filestr为读入的文件字节序。

加密：

使用密钥扩展算法得到加密密钥

SMS4KeyExt( (muint8 \*)key, rk, ENCRYPT);

用加密密钥加密得到密文c

SMS4Crypt(filestr,c,rk);

使用密钥扩展算法得到解密密钥

SMS4KeyExt( (muint8 \*)key, rk, DECRYPT);

用解密密钥解密得到明文

SMS4Crypt(c,filestr,rk);