# SM4加密算法实验报告

## SM4介绍

SM4算法是一种对称加密算法，也被称为国密算法。它是由中国密码学家设计的，已被列入国家密码局的标准。

SM4算法使用128位的密钥和分组大小，使用32轮迭代加密，可以用于加密数据和验证消息认证码。它的加密效率很高，安全性也很好，被广泛应用于各种安全领域，如电子商务、移动通信和云计算等。

## 算法图解

### 加密算法

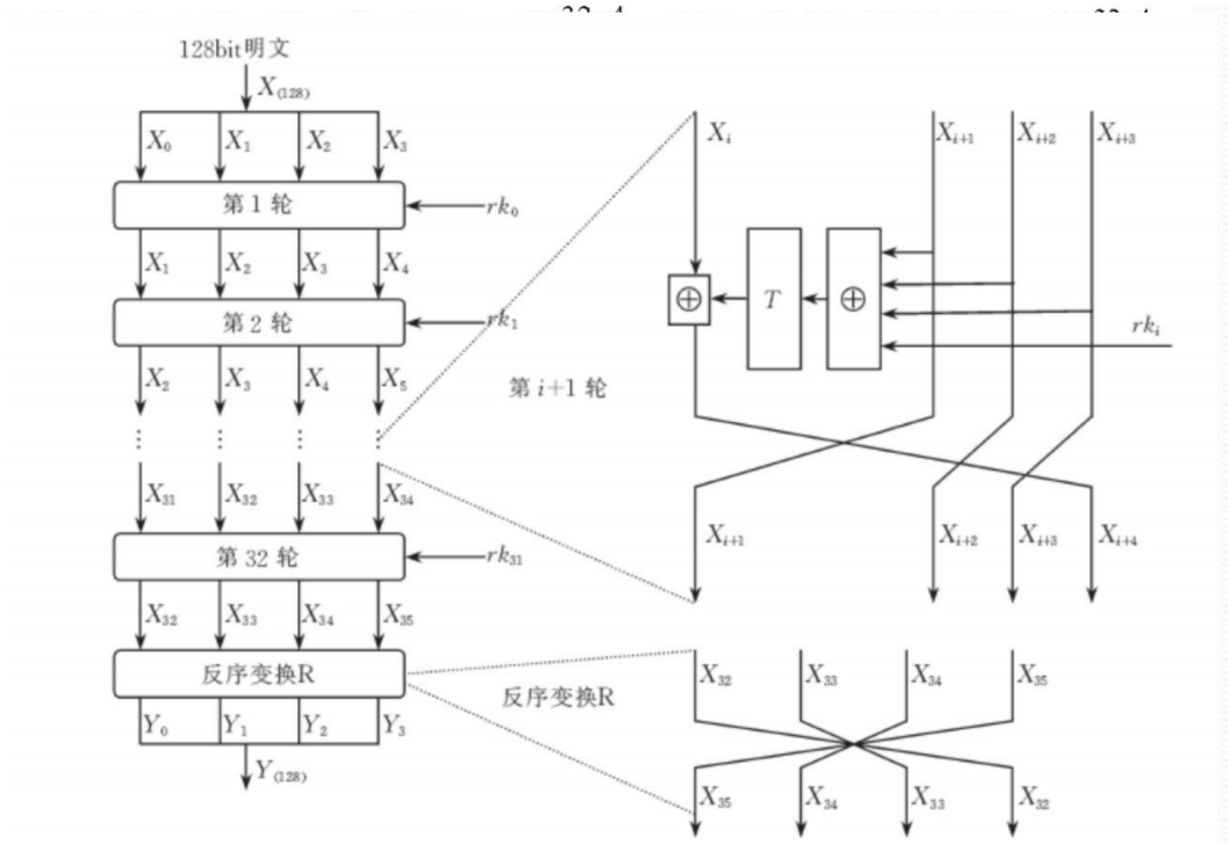
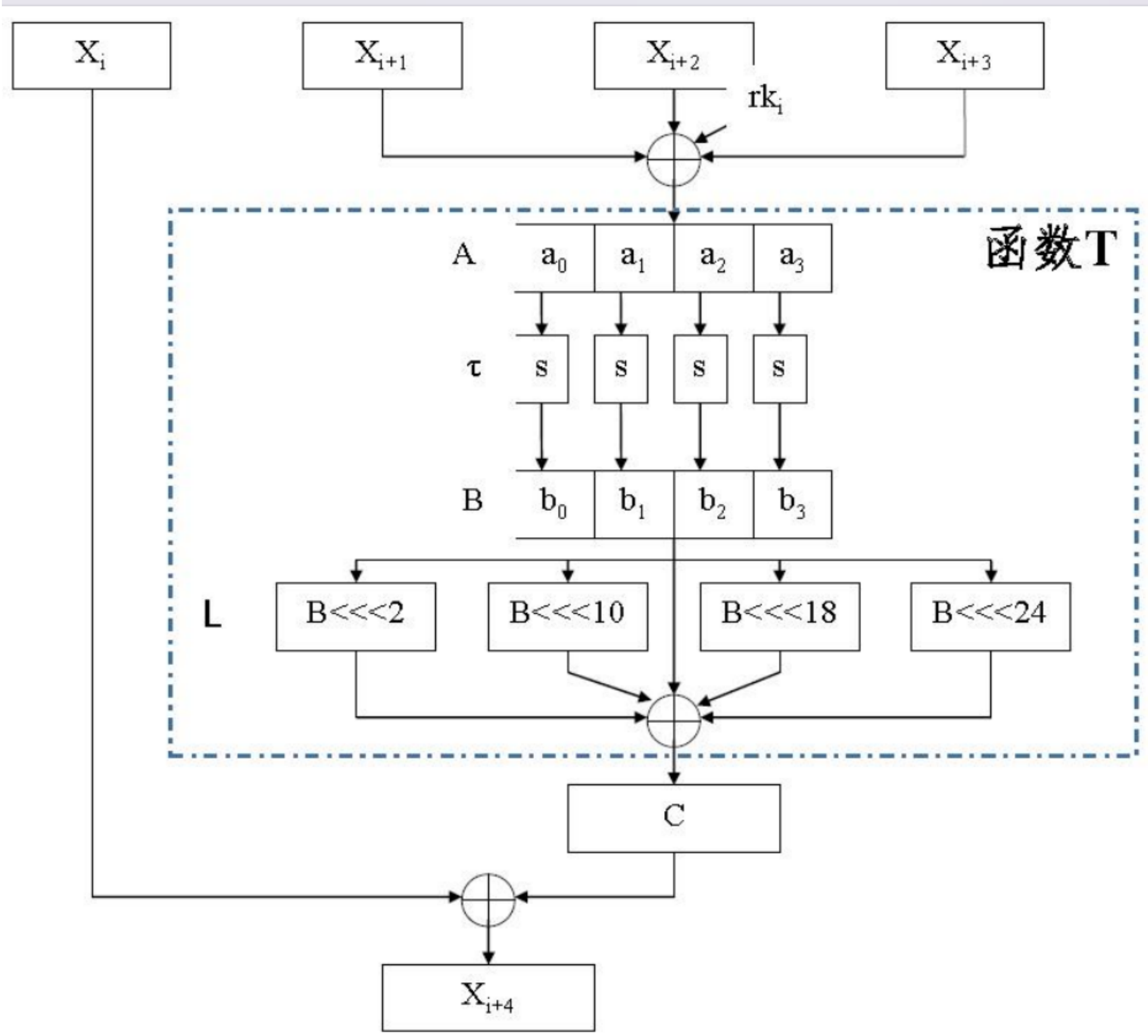
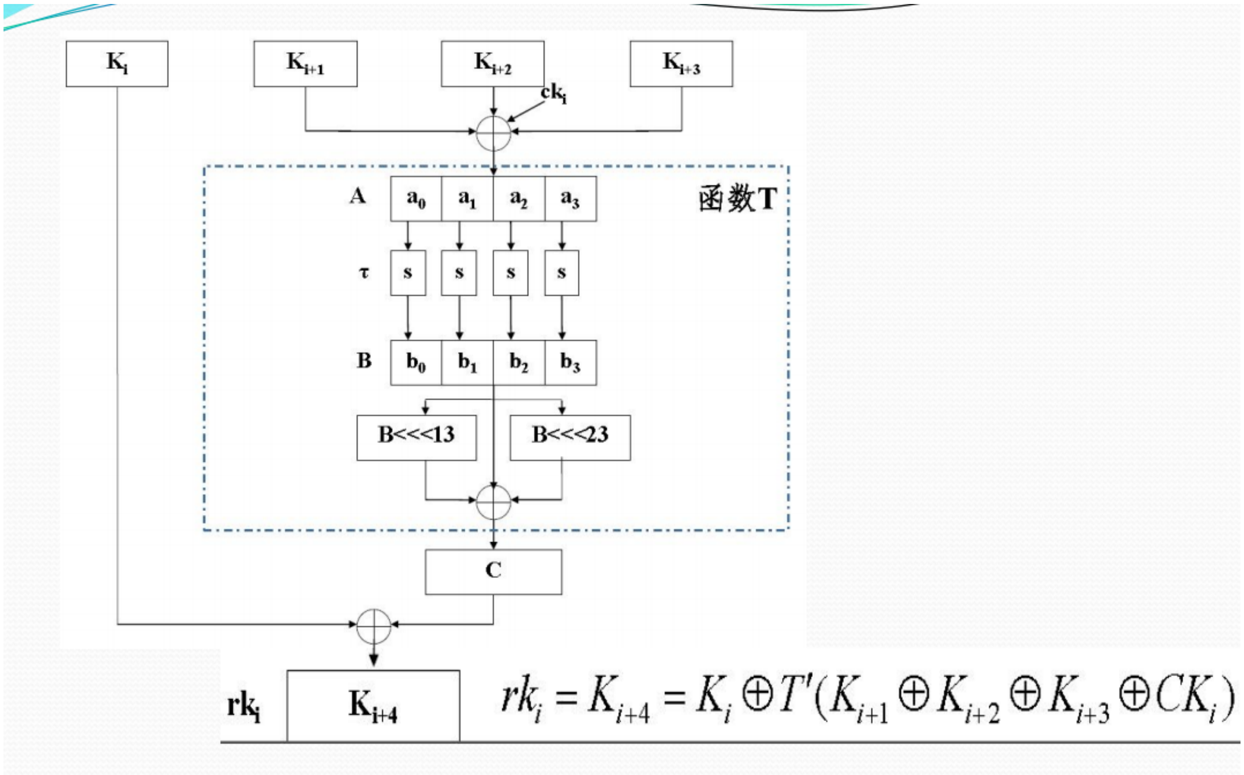



图 1 SMS4 加密算法整体结构



X_i
X_{i+1}
X_{i+2}
X_{i+3}
rk_i
函数T
A
a_0
a_1
a_2
a_3
τ
s
s
s
s
B
b_0
b_1
b_2
b_3
L
B<<<2
B<<<10
B<<<18
B<<<24
C
X_{i+4}


**密钥扩展算法**




## 重要代码解释

### 默认参数： FK , S盒以及CK

```
FK=[0xa3b1bac6, 0x56aa3350, 0x677d9197, 0xb27022dc]
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
        ]
CK = [
    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
]
```

## 实现某个小功能的函数

```
#32bits字拆开为字节
def wd_to_bys(wd, bys):
bys.extend([(wd >> i) & 0xff for i in range(24, -1, -8)])

#把4字节合并为32bits字
def bys_to_wd(bys):
ret = 0
for i in range(4):
bits = 24 - i * 8
ret |= (bys[i] << bits)
return ret

#循环左移
def left(wd,bit):
return (wd<<bit & 0xffffffff) |(wd>>(32-bit ))

#循环右移
def right(wd,bit):
return (wd>>bit & 0xffffffff) | (wd<<(32-bit))

#查S盒,输入为32bits
def search_s(wd):
ret = []
for i in range(0, 4):
byte = (wd >> (32 - (i + 1) * 8)) & 0xff
row = byte >> 4
col = byte & 0x0f
index = (row * 16 + col)
ret.append(S_BOX[index])
```

```
return bys_to_wd(ret)

# T变换

def T(x1,x2,x3,rk):
a=x1^x2^x3^rk
b=search_s(a)
return b^left(b,2)^left(b,10)^left(b,18)^left(b,24)

#T'变换
def rT(k1,k2,k3,ck):
a=k1^k2^k3^ck
b=search_s(a)
return b^left(b,13) ^ left(b,23)

#逆向
def rever(x):
for i in range(4):
x[3-i]= (x[3-i] & 0xffffffff)
s = f"{x[3]:08x}{x[2]:08x}{x[1]:08x}{x[0]:08x}"

return s
#输出
def output(s, name):

•    out = ""
•    for i in range(0, len(s), 2):
•        out += s[i:i + 2] + " "
•    print(f"{name}:", end="")
•    print(out.strip())
```

## 密钥扩展算法

```
def extend(mk):

  """

  密钥扩展算法

  """

  MK=[(mk >> (128 - (i + 1) * 32)) & 0xffffffff for i in range(4)]  #分割为8bits
一组

  K=[MK[i] ^ FK[i] for i in range (4)]

  rk=[]

  for i in range(32):     #生成轮密钥

```

```
        a=rT(K[i+1],K[i+2],K[i+3],CK[i])

        K.append(K[i]^a)

        rk.append(K[i]^a)

    return rk
```

### 加密算法

```
#####加密算法#####

def encode(x,rk):

    X=[(x >> (128 -(i + 1) *32) )& 0xffffffff  for i in range(4)]   ##分割



    for i in range(32):     #轮函数加密

        c=T(X[1],X[2],X[3],rk[i]) ^ X[0]

        X=X[1:]+[c]

    Y=rever(X)        #将得到的X逆向

    return  Y
```

### 解密算法

```
#####解密算法#####

def decode(ciphertext, rk):    #操作几乎相同
    ciphertext = int(ciphertext, 16)
    X = [ciphertext >> (128 - (i + 1) * 32) & 0xffffffff for i in range(4)]
    for i in range(32):
        t = T(X[1], X[2], X[3], rk[31 - i])
        c = (t ^ X[0])
        X = X[1:] + [c]
    m = rever(X)
    return m
```

## 简单测试

首先试运行国标文件上的示例

本附录为 SM4 分组密码算法对一组明文进行加密的运算示例。

**输入明文：** 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

**输入密钥：** 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

**输出密文：** 68 1E DF 34 D2 06 96 5E 86 B3 E9 4F 53 6E 42 46

运行的代码以及运行结果

```
x=0x0123456789abcdeffedcba9876543210
mk=0x0123456789abcdeffedcba9876543210
rk=extend(mk)
ciphertext=encode(x,rk)
output(ciphertext,"cipertext")
plaintext=decode(ciphertext,rk)
output(plaintext,"plaintext")
```

```
ons\ms-python.debugpy-2024.12.0-win32-x64\bundled\libs\debugpy\a
s\86199\Desktop\保密技术基础\Cryptography-lab-\SM4\SM4.py'
cipertext:68 1e df 34 d2 06 96 5e 86 b3 e9 4f 53 6e 42 46
plaintext:01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
```

然后加密我自定义的内容

```
x=0x202202010212 6c756f6265696e692004
mk=0x2022030302127a6f756a696168616f02
```

结果:

```
cipertext:08 97 fd ca 28 83 cb 99 15 04 61 40 07 2e 9b 9f
plaintext:20 22 02 01 02 12 6c 75 6f 62 65 69 6e 69 20 04
```