SM4 加密算法实现与查找表优化实验报告

1. 实验目标

1. 实现国密 SM4 分组密码算法的加密/解密功能
2. 验证算法实现的正确性和安全性
3. 通过查找表优化技术提升算法执行效率

2. 算法原理

2.1 SM4 算法概述

SM4 是中国国家密码管理局发布的分组密码标准，主要特性包括：

分组长度：128 位

密钥长度：128 位

轮数：32 轮非平衡 Feistel 结构

3. 关键实现解析

3.1 查找表预计算

设计原理：

将 S 盒替换和线性变换合并计算，为每个字节位置独立建表：

Table0：高位字节变换结果

Table1：次高位字节变换结果

Table2：次低位字节变换结果

Table3：低位字节变换结果

```
static void SM4_inner(uint8_t* in, uint8_t* out, SM4_Key sm4_key, uint8_t enc) {
    // 数据加载（128→32×4）
    for(int i=0; i<4; i++){
        x[i] = (in[4*i]<<24) | (in[4*i+1]<<16) | (in[4*i+2]<<8) | in[4*i+3];
    }

    // 32 轮迭代（查表优化核心）
```

```
    for(int i=0; i<32; i++){
        tmp = (enc ? sm4_key[31-i] : sm4_key[i]); // 轮密钥选择
        tmp = x[1] ^ x[2] ^ x[3] ^ tmp;          // 轮密钥混合

        // 四路并行查表（消除计算依赖）
        tmp1 = Table0[tmp >> 24];          // 高位字节变换
        tmp1 ^= Table1[(tmp>>16)&0xFF];    // 次高位字节变换
        tmp1 ^= Table2[(tmp>>8)&0xFF];     // 次低位字节变换
        tmp1 ^= Table3[tmp&0xFF];          // 低位字节变换
        tmp1 ^= x[0];                      // Feistel 结构混合

        // 寄存器移位
        x[0]=x[1]; x[1]=x[2]; x[2]=x[3]; x[3]=tmp1;
    }

    // 结果重组（32×4→128）
    for(int i=0; i<4; i++){
        uint32_t w = x[3-i]; // 反序输出
        out[4*i] = w>>24; out[4*i+1] = w>>16;
        out[4*i+2] = w>>8; out[4*i+3] = w;
    }
}
```

相比较查表方法的优点：

计算预合并：

Ti[b]=SBox(b)⊕(SBox(b)<<2)⊕(SBox(b)<<10)⊕(SBox(b)<<18)⊕(SBox(b)<<24)

并行查表：4 个字节独立查表

消除依赖链：解决移位操作的顺序依赖

可以明显减少时钟周期。

正确性检验：

```
Encrypted: 44 92 f0 a8 ae 3c cd b6 b5 d9 dc b8 21 af da f4
Decrypted: 02 21 45 47 89 ab cd ef be de ba 08 70 58 11 ae

Time for encryption of SM4: 0.0000009 s
```

具有加解密一致性，基本正确。