Ch3 分组密码

（本讲主要介绍DES和AES的算法和详细过程）

DES的明文、密文和密钥的分组长度都是64位，面向二进制数据，综合运用了置换、代替、代数等基本密码技术，加密和解密共用了同一个算法，其基本密码结构属于Feistel结构。

DES

DES算法框图

```
graph TB
    A[64位明文]-->B[IP]
    B-->G1([L0])
    B-->G2([R0])
    G2-->G3([L1])
    G1-->G4([R1])
    G2-->H
    H[轮函数F]
    I[某种异或运算+]
    H-->I
    G1-->I
    I-->G4
    E[64位密钥]-->子密钥产生-->F(子密钥K1,K2,...,K10)-->D[(K1)]
    D-->H
    K[...]
    G3-->K
    G4-->K
    K-->L[逆初始置换]
    L-->64位输出
```

加密的迭代过程如下：
$R_i = L_{i-1} \oplus F(R_{i-1}, K_i), L_i = R_{i-1}$

解密的迭代过程如下：
$R_{i-1}=L_{i} \oplus F(L_{i}, K_{i-1}), R_{i-1} = L_{i}$

关于加密函数（轮函数）f：保密的核心

```
graph TB
    A[32位输入]-->B[选择运算K]
    B-->C[48位中间结果]
    D([异或运算])
    C-->D
    E[48位子密钥]
    E-->D
    S([代替函数组S-box])
    S-->F4
    D-->F1[S1]
```

```
    D-->F2[S2]
    D-->F3[S3...]
    D-->F4[S8]
    G(置换运算)
    F1-->G
    F2-->G
    F3-->G
    F4-->G
    G-->32位输出
```

DES具有可逆性和对合性，

DES的安全性？？

DES是经不起穷举攻击的。

- DES挑战赛

```
msg = "The unknown message is: XXXX"
CT =
```

目标：给定$(m_i, c_i = E(k, m_i)), i=1, 2, 3$，找出密钥$k∈{0,39}^{56}$

- 安全弱点（最根本在于）：密钥太短，只有56个有效比特

【由此开发了3DES，112个有效比特】

- 存在弱密钥和半弱密钥：在每次迭代时都有一个子密钥供加密使用，如果给定初始密钥${k}$，由于各轮的子密钥都相同，那么就可以计算出所有的${k}$。半弱密钥则是指由给定k产生的所有密钥中有不完全相同的重复者。
- 存在互补对称性，如果两个密钥互补的结果等于0，那么这两个密钥是对称的。

AES 是 128 位分组加密算法，其密钥长度为 128、192、256 位，采用了【混淆】和【扩散】两大特性。

S盒的设计准则——混淆；P盒的设计准则——扩散

AES的数据处理方式：

- 按字节处理
- 按字处理
- 按状态处理

AES的数据处理过程图示： $$ \begin{array}{ll} x=x_0x_1x_2...x_{127} \end{array}= \begin{pmatrix} x_{0}&x_{1}&x_{2}&x_{3} \ x_{4}&x_{5}&x_{6}&x_{7} \ x_{8}&x_{9}&x_{10}&x_{11} \ x_{12}&x_{13}&x_{14}&x_{15} \end{pmatrix}= \begin{pmatrix} s_{00}&s_{01}&s_{02}&s_{03} \ s_{10}&s_{11}&s_{12}&s_{13} \ s_{20}&s_{21}&s_{22}&s_{23} \ s_{30}&s_{31}&s_{32}&s_{33} \end{pmatrix}= \begin{pmatrix}

\end{pmatrix} $$

AES的算法结构：

详细介绍

初始轮变换（非线性层 S盒变换 ByteSub/State）

- S盒是AES中的唯一一个非线性变换，是AES安全的关键
- 核心运算：$GF(2^8)$——用多项式 $a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x^1+a_0x^0$表示比特位的有限域
  - 将输入字节用$GF(2^8)$上的逆来替代
  - 倒序+仿射变换

线性混合层（行移位变换 ShiftRow、列混合变换 MixColumn）

- 第0行不移位，第一行移动C1字节，第二行移动C2字节，第三行移动C3字节 $$ \begin{pmatrix} s_{00}&s_{01}&s_{02}&s_{03}\\ s_{10}&s_{11}&s_{12}&s_{13}\\ s_{20}&s_{21}&s_{22}&s_{23}\\ s_{30}&s_{31}&s_{32}&s_{33}\\ \end{pmatrix}\Longrightarrow \begin{pmatrix} s_{00}&s_{01}&s_{02}&s_{03}\\ s_{11}&s_{12}&s_{13}&s_{10}\\ s_{21}&s_{22}&s_{20}&s_{21}\\ s_{32}&s_{30}&s_{31}&s_{32}\\ \end{pmatrix} $$
- 将状态的每列视为$GF(2^8)$上的多项式$a(x)$，乘上一个多项式$c(x)$（与x^4+1互素），模x^4+1，即 $$\begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3 \end{pmatrix}= \begin{pmatrix} b_{00}&b_{01}&b_{02}&b_{03}\\ b_{10}&b_{11}&b_{12}&b_{13}\\ b_{20}&b_{21}&b_{22}&b_{23}\\ b_{30}&b_{31}&b_{32}&b_{33}\\ \end{pmatrix} \begin{pmatrix} 03\\ 01\\ 01\\ 02 \end{pmatrix} $$ 密钥加层（AddRoundKey）【注意：最后一轮中没有密钥加变换】

参考代码实现

```
const int NR = 10;
const int NC = 4;
bitset<8> preprocess(array<bitset<32>, 4> bytes) {
    bitset<8> wordres = 0x00000000, temp;
    for (int i = 0; i < 4; i++) {
        temp = bytes[i].to_ulong();
        temp <<= (3 - i) * 8;
        wordres |= temp;
    }
    return wordres;
}
bitset<8> subword(bitset<32> sw) {
    word temp;
    for (int i = 0; i < 32; i += 8) {
        int row = sw[i + 7] * 8 + sw[i + 6] * 4 + sw[i + 5] * 2 + sw[i +
4];
        int col = sw[i + 3] * 8 + sw[i + 2] * 4 + sw[i + 1] * 2 + sw[i];
        byte tempvar = SBox[row * 16 + col];
        for (int j = 0; j < 8; j++) {
            temp[i + j] = tempvar[j];
        }
    }
    return temp;
}
```