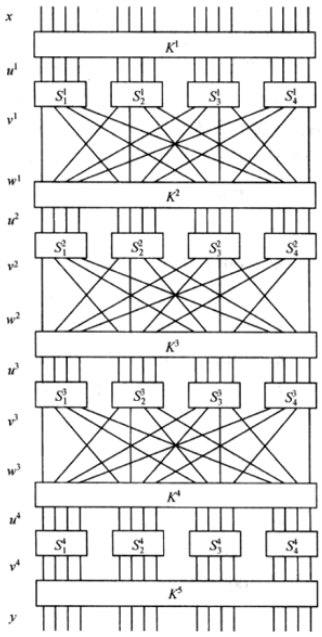
SPN 线性攻击

姓名：张刘明　　学号：2110049

实验要求：

要求大家实现线性攻击（P. 68-69）算法；　分析出 K^5 轮密钥…



SPN 网络示意图

实验过程：

相关概念：

SPN 线性密码分析:

是一种基于 S 盒逼近的分析方法，需要已知明文以及较多的明密文对，SPN 先行密码分析算法只能分析最后一轮子密钥，缩小了密钥穷举范围，但是当使用该方法分析出最后一轮密钥时，由于密钥生成算法固定，为分析第一轮密钥提供了可能。

堆积引理:

对取值于{0, 1}上的随机变量，用分布偏差来表示它的概率分布常常是很方便的。$\mathbf{X}_i$ 的偏差被定义为：

$$\epsilon_i = p_i - \frac{1}{2}$$

注意下列事实：对 $i = 1, 2, \cdots$

$$-\frac{1}{2} \leqslant \epsilon_i \leqslant \frac{1}{2}$$

$$\Pr[\mathbf{X}_i = 0] = \frac{1}{2} + \epsilon_i$$

$$\Pr[\mathbf{X}_i = 1] = \frac{1}{2} - \epsilon_i$$

引理 3.1(堆积引理) 设 $\mathbf{X}_{i_1}, \cdots, \mathbf{X}_{i_k}$ 是独立随机变量，$\epsilon_{i_1, i_2, \cdots, i_k}$ $(i_1 < i_2 < \cdots < i_k)$ 表示随机变量 $\mathbf{X}_{i_1} \oplus \mathbf{X}_{i_2} \oplus \cdots \oplus \mathbf{X}_{i_k}$ 的偏差，则

$$\epsilon_{i_1, i_2, \cdots, i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}$$

S 盒的线性逼近

如果

$(y_1, \cdots, y_n) \neq \pi_S(x_1, \cdots, x_m)$，则

$$\Pr[\mathbf{X}_1 = x_1, \cdots, \mathbf{X}_m = x_m, \mathbf{Y}_1 = y_1, \cdots, \mathbf{Y}_n = y_n] = 0$$

如果 $(y_1, \cdots, y_n) \neq \pi_S(x_1, \cdots, x_m)$，则

$$\Pr[\mathbf{X}_1 = x_1, \cdots, \mathbf{X}_m = x_m, \mathbf{Y}_1 = y_1, \cdots, \mathbf{Y}_n = y_n] = 2^{-m}$$

如果 $(y_1, \cdots, y_n) = \pi_S(x_1, \cdots, x_m)$，则

$$\Pr[\mathbf{Y}_1 = y_1, \cdots, \mathbf{Y}_n = y_n \mid \mathbf{X}_1 = x_1, \cdots, \mathbf{X}_m = x_m] = 1$$

线性分析原理:

线性分析过程:

SPN 的线性分析过程需要首先找出一组 S 盒的线性逼近，用于导出一整个 SPN 的线性逼近
本次实验中的逼近包括如下的四个活动 S 盒:

- 在 S_2^1 中，随机变量 $T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1$ 具有偏差 1/4
- 在 S_2^2 中，随机变量 $T_2 = U_6^2 \oplus V_6^2 \oplus V_8^2$ 具有偏差−1/4
- 在 S_2^3 中，随机变量 $T_3 = U_6^3 \oplus V_6^3 \oplus V_8^3$ 具有偏差−1/4
- 在 S_4^3 中，随机变量 $T_4 = U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3$ 具有偏差−1/4

因此，我们可以得出线性分析的过程如下：

1）收集大量明密文对；

2）选择一个固定的输入位，以便构建 S 盒线性逼近；

$$in(x,y) = (\sum_{i=1}^{4} \oplus a_i x_i) \oplus (\sum_{i=1}^{4} \oplus b_i y_i),$$

3）构建线性逼近表：通过收集的明密文对，根据上面构造的 in(x,y)构建线性逼近表，记录输入位和输出位之间的线性关系；

4）构建线性逼近链：根据输入，选取线性逼近表中偏差最大的作为输出，找到对应位置进行输入；再根据线性逼近表找到输出；以此类推，构建出线性逼近链（偏差越大，越具有线性关系）；

5）化简关系式：根据该线性逼近链，通过将中间过程中的输入输出相异或来化简过程，最终化简为只剩下输入和最后一轮输入异或的关系式 test()

6）遍历密钥和明文密文对：遍历所有可能的密钥，遍历所有收集到的明密文对，通过 y 和当前测试的密钥计算出最后一轮输入，将该值带入关系式 test()中，若该关系式为 0，则该密钥对应 count 值应该加一；

7）输出最可能的密钥：遍历结束后输出 count 值最大的密钥

SPN 线性分析伪代码:

算法 3.2　线性攻击$(\mathcal{T}, T, \pi_S^{-1})$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

　do $\text{Count}[L_1, L_2] \leftarrow 0$

for each $(x, y) \in \mathcal{T}$

do { **for** $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

do { $v_{<2>}^4 \leftarrow L_1 \oplus y_{<2>}$

$v_{<4>}^4 \leftarrow L_2 \oplus y_{<4>}$

$u_{<2>}^4 \leftarrow \pi_S^{-1}(v_{<2>}^4)$

$u_{<4>}^4 \leftarrow \pi_S^{-1}(v_{<4>}^4)$

$z \leftarrow x_5 \oplus x_7 \oplus x_8 \oplus u_6^4 \oplus u_8^4 \oplus u_{14}^4 \oplus u_{16}^4$

if $z = 0$

　then $\text{Count}[L_1, L_2] \leftarrow \text{Count}[L_1, L_2] + 1$

$\max \leftarrow -1$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

do { $\text{Count}[L_1, L_2] \leftarrow |\text{Count}[L_1, L_2] - T/2|$

if $\text{Count}[L_1, L_2] > \max$

then { $\max \leftarrow \text{Count}[L_1, L_2]$

$\text{maxkey} \leftarrow (L_1, L_2)$

output(maxkey)

代码:

```
#include<stdio.h>
#include<iostream>
```

```
unsigned short sbox[16] =
{ 0xe,0x4,0xd,0x1,0x2,0xf,0xb,0x8,0x3,0xa,0x6,0xc,0x5,0x9,0x0,0x7 };//s盒
unsigned short pbox[16] = { 1,5,9,13,2,6,10,14,3,7,11,15,4,8,12,16 };//p盒
unsigned short vsbox[16] =
{ 0xe,0x3,0x4,0x8,0x1,0xc,0xa,0xf,0x7,0xd,0x9,0x6,0xb,0x2,0x0,0x5 };//s盒逆盒
unsigned short vpbox[16] = { 1,5,9,13,2,6,10,14,3,7,11,15,4,8,12,16 };//p盒逆置换

int read();//快速读入
void spn(unsigned int key, unsigned short x);
void roundkeys(int i, unsigned short& k, unsigned int key);//获取轮密钥
void sreplace(unsigned short u, unsigned short& v);//s盒代换
void preplace(unsigned short v, unsigned short& w);//p盒置换
void vsreplace(unsigned short v, unsigned short& u);//s盒逆代换
void vpreplace(unsigned short w, unsigned short& v);//p盒逆置换

int main()
{
    int i, n;
    unsigned int key;
    unsigned short x;
    std::cin >> n;
    getchar();
    for (i = 0; i < n; i++) {
        key = read();
        x = read();
        spn(key, x);
    }
}

int read() {//快速读入
    char ch;
    int i = 0;
    int x = 0;
    ch = getchar();
    while (ch != ' ' && ch != '\n')
    {
        x *= 16;
        if (ch >= '0' && ch <= '9') x += ch - '0';
        else x += ch - 'a' + 10;
        ch = getchar();
    }
    return x;
}
```

```
void spn(unsigned int key, unsigned short x)
{
    unsigned short w = x;
    unsigned short k, u, v;
    unsigned short y;
    int n = 4;
    int i;
    for (i = 1; i <= n - 1; i++) {//三轮加密
        roundkeys(i, k, key);
        u = k ^ w;
        sreplace(u, v);
        preplace(v, w);
    }
    roundkeys(n, k, key);
    u = k ^ w;
    sreplace(u, v);
    roundkeys(n + 1, k, key);
    y = v ^ k;
    printf("%04x ", y);

    y ^= 0x1;
    v = y ^ k;
    vsreplace(v, u);
    roundkeys(n, k, key);
    w = u ^ k;
    for (i = n - 1; i >= 1; i--)//三轮解密
    {
        vpreplace(w, v);
        vsreplace(v, u);
        roundkeys(i, k, key);
        w = u ^ k;
    }
    x = w;
    printf("%04x\n", x);
}

void roundkeys(int i, unsigned short& k, unsigned int key)//获取轮密钥
{
    unsigned int temp = 0xffff0000;
    i -= 1;
    temp = temp >> (4 * i);
    k = (key & temp) >> (4 - i) * 4;
}
```

```
void sreplace(unsigned short u, unsigned short& v)//s盒代换
{
    unsigned short temp[] = { 0xf000,0x0f00,0x00f0,0x000f };
    unsigned short j, t = 0x0000;
    int i;
    for (i = 0; i < 4; i++) {
        j = (u & temp[i]) >> (4 * (3 - i));
        t |= (sbox[j] << (4 * (3 - i)));
    }
    v = t;
}

void preplace(unsigned short v, unsigned short& w)//p盒置换
{
    unsigned int temp = 0x10000;
    unsigned short t, j = 0x0000;
    int i;
    for (i = 0; i < 16; i++) {
        temp = temp >> 1;
        t = (v & temp) >> (15 - i);
        j |= (t << (16 - pbox[i]));
    }
    w = j;
}

void vsreplace(unsigned short v, unsigned short& u)//s盒逆代换
{
    unsigned short temp[] = { 0xf000,0x0f00,0x00f0,0x000f };
    unsigned short j, t = 0x0000;
    int i;
    for (i = 0; i < 4; i++) {
        j = (v & temp[i]) >> (4 * (3 - i));
        t |= (vsbox[j] << (4 * (3 - i)));
    }
    u = t;
}

void vpreplace(unsigned short w, unsigned short& v)//p盒逆置换
{
    unsigned int temp = 0x10000;
    unsigned short t, j = 0x0000;
    int i;
    for (i = 0; i < 16; i++) {
```

```
            temp = temp >> 1;
            t = (w & temp) >> (15 - i);
            j |= (t << (16 - vpbox[i]));
        }
        v = j;
}
```

Github仓库地址：https://github.com/newstarming/Crypto.git