2110688 史文天 信息安全专业 第一次实验报告

线性攻击算法

（一）线性攻击算法实现

- 通过Python脚本随机生成16000个16位二进制串，将其放入SPN加密算法中，以`0011101010010100110101100011111 1`为密钥进行加密得到密文，然后将明密文对写入`pairs.txt`文件，得到本次实验的数据集。该密钥K_5轮密钥为`1101011000111111`

```
import subprocess  # 导入subprocess模块，用于执行外部命令
import random  # 导入random模块，用于生成随机数据

# 生成指定长度的随机二进制字符串
def generate_random_binary_string(length=16):
    return ''.join(random.choice('01') for _ in range(length))

executable_file_path = "./SPN.exe"  # 定义外部可执行文件的路径
num_pairs = 16000  # 定义要生成的数据对数

# 打开文件以写入生成的数据对，"pairs.txt"是文件名
with open("./data/pairs.txt", "w") as file:
    for _ in range(num_pairs):  # 生成指定数量的数据对
        input_data = generate_random_binary_string()  # 生成随机的输入数据

        # 运行外部可执行文件并将输入数据传递给它
        result = subprocess.run([executable_file_path],
input=input_data.encode(), stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        output_data = result.stdout.decode().strip()  # 获取外部命令的标准输出

        # 将生成的输入数据和外部命令的输出数据写入文件
        file.write(input_data + "\n")
        file.write(output_data + "\n")
```

- 全局部分定义了`S`盒、明文与密文数组、计数器，以及Dec2Bin（十进制转二进制）的函数。

```
int S[16] = { 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7 };// S盒
int x[16] = {};// 明文
int y[16] = {};// 密文
int Count[16][16] = {};// 计数器
// 将十进制数转换为二进制表示
void Dec2Bin(int decimal, int* binary, int num){
    int i = num - 3;
    while (decimal > 0){
        binary[num] = decimal % 2;
        decimal /= 2;
        num--;
    }
    while (num >= i){
        binary[num] = 0;
        num--;
```

```
    }
}
```

- 下面便是`main`函数部分。
  - 遍历`(0,0) to (F,F)`，按照算法内容进行计算，得到对应的计数器。

```
int S1[16] = {};
for (int i = 0; i < 16; i++){
    S1[S[i]] = i;
}

//读入数据集
ifstream input(".\\data\\pairs.txt");

// 遍历(0,0) to (F,F)，按照算法内容进行计算，得到对应的计数器
int L1[4] = {};
int L2[4] = {};
int v[16] = {};
int u[16] = {};

int n = 0;
cout << "Please enter the number of quadruple pairs to be analyzed: " <<
endl;
cin >> n;

auto start = high_resolution_clock::now();

for (int i = 0; i < n; i++)
{
    string X, Y;
    input >> X >> Y;

    int len = X.length();
    for (int j = 0; j < len; j++)
        x[j] = X[j] - '0';

    len = Y.length();
    for (int j = 0; j < len; j++)
        y[j] = Y[j] - '0';

    for (int j = 0; j < 16; j++){
        for (int k = 0; k < 16; k++){
            // 枚举 L1 和 L2 的可能取值
            DecimalToBinary(j, L1, 3);
            DecimalToBinary(k, L2, 3);

            // 计算 v 数组的元素
            v[4] = L1[0] ^ y[4];
            v[5] = L1[1] ^ y[5];
            v[6] = L1[2] ^ y[6];
            v[7] = L1[3] ^ y[7];

            v[12] = L2[0] ^ y[12];
            v[13] = L2[1] ^ y[13];
            v[14] = L2[2] ^ y[14];
```

```
            v[15] = L2[3] ^ y[15];

            // 对 v 应用 S1 变换，得到 u 数组的元素
            int temp1 = v[4] * pow(2, 3) + v[5] * pow(2, 2) + v[6] *
pow(2, 1) + v[7] * pow(2, 0);
            int temp2 = S1[temp1];
            DecimalToBinary(temp2, u, 7);

            temp1 = v[12] * pow(2, 3) + v[13] * pow(2, 2) + v[14] *
pow(2, 1) + v[15] * pow(2, 0);
            temp2 = S1[temp1];
            Dec2Bin(temp2, u, 15);

            // 计算 z 值
            int z = x[4] ^ x[6] ^ x[7] ^ u[5] ^ u[7] ^ u[13] ^ u[15];

            // 如果 z 等于0，增加 Count[j][k] 计数
            if (z == 0) Count[j][k]++;
        }
    }
}

auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);

input.close();
```

- 最后分析比较计数器的值，输出最大可能的轮密钥。

```
int max = -1;
int LL1 = 0, LL2 = 0;//记录最大的Count[i][j]对应的L1和L2
for (int i = 0; i < 16; i++)
{
    for (int j = 0; j < 16; j++)
    {
        Count[i][j] = abs(Count[i][j] - n / 2);
        if (Count[i][j] > max)
        {
            max = Count[i][j];
            LL1 = i;
            LL2 = j;
        }
    }
}

cout << "maxkey:" << endl;
Dec2Bin(LL1, L1, 3);
for (int i = 0; i < 4; i++)
{
    cout << L1[i];
}
cout << " ";
Dec2Bin(LL2, L2, 3);
for (int i = 0; i < 4; i++)
{
    cout << L2[i];
```

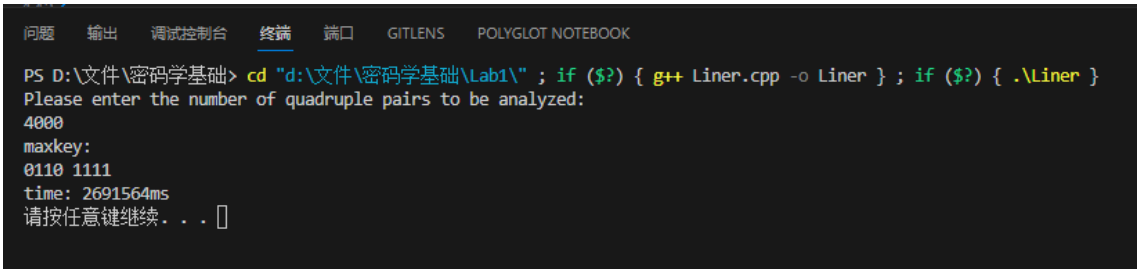
```
}

cout << endl << "time: " << duration.count() << "ms" << endl;
```

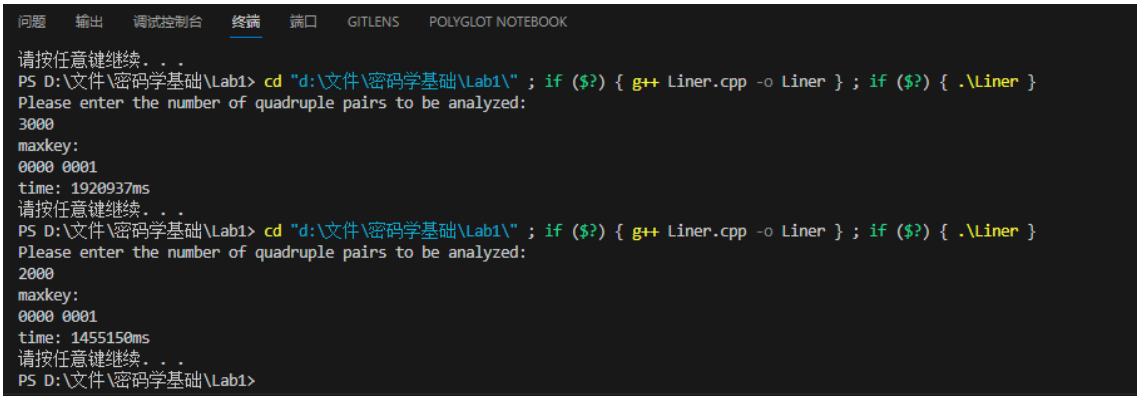
（二）密钥分析

该密钥K_5轮密钥为`1101011000111111`，则$L_1=0110$，$L_2=1111$。

- 首先尝试用4000对明密文进行攻击，成功得到密钥，用时2691ms。



- 经过测试，继续减少数据量则无法正确得到密钥。



- 修改传入密钥，得到新的数据集，测试攻击成功率。
  - 密钥为
    - `00000000000000000000000000000000`
      - 其轮密钥为`00000000`
    - `11111111111111111111111111111111`
      - 其轮密钥为`11111111`
    - `01010101010101010101010101010101`
      - 其轮密钥为`01010101`
    - `10101010101010101010101010101010`
      - 其轮密钥为`10101010`
  - 经过测试，当明密文对数为4000时成功率已经足够高，以上五组攻击均能成功。

- 在2得到第2、4部分轮密钥后，对于剩余的八位轮密钥进行分析。
  - 选择新的线性分析链，在第2、4部分密钥已知的基础上分析出第1、3部分的密钥。接着在已知起始密钥低16位的基础上，穷举高16位密钥对给出的8000个明密文对进行验证。由于在加密过程中进行了5轮的S代换和P置换，导致相同明文在不同密钥下得到相同密文的概率极低，因此在实际验证过程中并不需要验证8000个明密文对是否对应，而仅需验证3个即可判断出密钥是否合适。

- 由于虽然可以选取到仅包含第5轮第1、3部分的线性分析链，但是由于偏差不大，因此代表性较差，可能导致对于1、3部分密钥可能性较大部分的遍历过多，而导致时间开销较大。