## **SPN线性密码分析实验报告**

曹瑜 2212794

1. 实验内容：

根据大量已知明密文对通过线性分析得出原始SPN算法中的的最后一轮密钥；

输入：8000个明密文对；

输出：最后一轮密钥；

1. 实验原理：

堆积引理 & S盒的线性逼近

**堆积引理**：设 、…… 是独立随机变量，（i1<i2<…<ik）表示随机变量 ⊕ ⊕ … ⊕ 的偏差，那么 = 2k-1 

**推论**：若对于某一个，有 = 0， 那么=0；即存在一个完全随机的变量时，总偏差为0；

基于堆积引理，可进行**S盒的线性逼近**：

选取若干输入变量和输出变量，分析这些随机变量异或得到的组合随机变量，通过穷举变量取值计算其偏差，偏差越大，说明选取的变量间线性关联越强，由此可确定256个组合异或随机变量的偏差值，将每个异或组合用两个二元向量（a1,a2,a3,a4）和（b1,b2,b3,b4）表示，

即： 

将（a1,a2,a3,a4）和（b1,b2,b3,b4）用十六进制数表示，a=（a1,a2,a3,a4），b=（b1,b2,b3,b4），可定义NL（a，b）为满足：

=0

及 （y1,y2,y3,y4）=πs(x1,x2,x3,x4)

的二进制8元组个数，则对应组合随机变量的偏差计算公式为：

（NL (a，b）-8)/16

据此可得出对应的线性逼近表，因此可找出一组S盒的线性逼近，用来导出整个SPN算法（除最后一轮外）的线性逼近，线性逼近过程中选取若干活动S盒进行线性逼近；

按教材例，选择如下4个活动S盒：

、、、

再根据线性逼近表选取对应盒中偏差值较大的4个随机变量：

对应偏差值为1/4 、-1/4、 -1/4、 1/4 ；

根据SPN原理进行代换及异或后可得四个变量异或值为：





假设密钥比特固定，则随机变量具有非0偏差值

此时找到最后一轮与活动盒

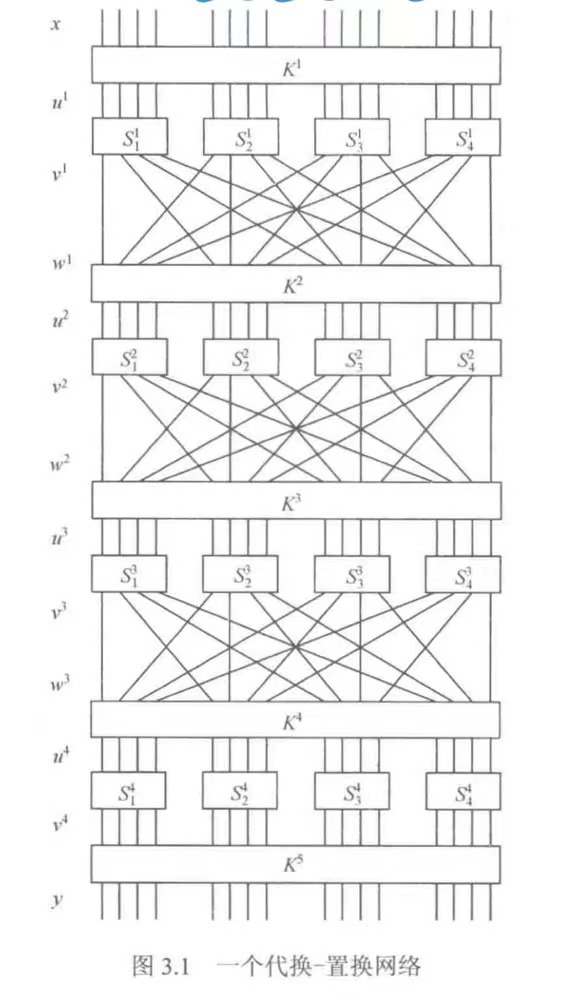
、 的输出相异或的子轮密钥

共256种可能，将其记为L1、L2 ，对于所有的明密文对，及所有的L1、L2取值，计算下式

 （1）

的值，若式值为0则给对应的L1、L2 的计数器加1，真正的候选子密钥count值应为T/2±T/32;

最后对8000个明密文对和256种子密钥组合全部进行遍历后，最大count值对应的子密钥即为真正的正确子密钥；



可知式（1）中只含有输入明文和第四轮S变换前的输入，通过已知密文Y可以反推出最后一轮S变换的输出，继而再推出该轮输入，即可计算出式（1）的值，进行计数；

式（1）值为0表示当前x、y有线性关联，当固定的L1、L2对于不同的x、y取值能对式（1）取到最多次0时，其为正确子密钥的可能性最大；

1. 算法设计：
2. 寻找s盒最好的线性逼近；
3. 分析活动s盒的变量变化路径；
4. 寻找具有较大偏差的随机变量组合；

穷举轮密钥的所有可能值，通过已知密文Y反推出最后一轮S变换的输出，继而再推出该轮输入，然后计算候选子密钥对应的变量组合式值并进行计数，选取最大计数器值对应的候选子密钥即为推测的正确子密钥

1. 代码实现：

// spn\_linear.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <vector>

#include <unordered\_map

using namespace std;

const int NUM\_PAIRS = 8000;

void substitution(int array[8], unordered\_map<string, string>& s\_rule) {

for (int i = 0; i < 8; i += 4) { // 每四个一组

string group;

for (int j = 0; j < 4; ++j) {

group += to\_string(array[i + j]); // 将整数转换为字符串

}

if (s\_rule.find(group) != s\_rule.end()) {

string substituted\_group = s\_rule[group];

for (int j = 0; j < 4; ++j) {

string substr = substituted\_group.substr(j, 1);

// 获取替换后的单个字符

array[i + j] = stoi(substr); // 将字符转换为整数并存回原数组

}

}

}

}

void convertToBinary(int num, int\* L1, int\* L2) {

for (int i = 7; i >= 0; --i) {

if (i >= 4) {

L1[i - 4] = (num >> i) & 1;

}

else {

L2[i] = (num >> i) & 1;

}

}

}

int main()

{

unordered\_map<string, string> s\_rule = {

{"1110", "0000"}, {"0100", "0001"}, {"1101", "0010"}, {"0001", "0011"},

{"0010", "0100"}, {"1111", "0101"}, {"1011", "0110"}, {"1000", "0111"},

{"0011", "1000"}, {"1010", "1001"}, {"0110", "1010"}, {"1100", "1011"},

{"0101", "1100"}, {"1001", "1101"}, {"0000", "1110"}, {"0111", "1111"}

};

ifstream inputFile("input.txt");

if (!inputFile) {

cerr << "无法打开文件！" << endl;

return 1;

}

vector<string> P(NUM\_PAIRS);

vector<string> C(NUM\_PAIRS);

int count[256]={0}; //计数器

// 逐行读取明文密文对

string line;

for (int i = 0; i < NUM\_PAIRS; ++i) {

getline(inputFile, line);

istringstream iss(line); //隔空格读取字符串

iss >> P[i];

iss >> C[i];

}

inputFile.close();

int L1[4]={0};

int L2[4]={0};

for (int i = 0; i < 8000; i++) {

int y[16]; //开始对p[i]、c[i]明密文对的计数

int x[16];

for (int j = 0; j < 16; ++j) {

// 将字符转换为数字并存储到数组中

x[j] = P[i][j] - '0';

y[j] = C[i][j] - '0';

}

// 生成所有可能的L1、L2取值组合

for (int a = 0; a < 256; ++a) {

// 将 a 转换为二进制，存储到 L1 和 L2 中

convertToBinary(a, L1, L2);

//对每一组L1 L2 开始最内层循环

int v[8];

for (int k = 0; k < 4; k++) {

v[k] = L1[k] ^ y[k + 4]; //v(盒2)=L1^y(盒2)

}

for (int k = 0; k < 4; k++) {

v[k+4] =L2[k] ^ y[k + 12]; //v(盒4)=L2^y(盒4)

}

substitution(v, s\_rule); //v（盒24）进行s逆变换

int z = x[4] ^ x[6] ^ x[7] ^ v[1] ^ v[3] ^ v[5] ^ v[7];

if (z == 0) {

count[a]++;

}

}

}

int max = 100;

int key = 0;

for (int i = 0; i < 256; i++) {

if (count[i] > max) {

max = count[i];

key = i;

}

}

cout <<"max count:" << max << endl;

convertToBinary(key, L1, L2);

cout << "key:";

for (int i = 0; i < 4; i++) {

cout << L1[i];

}

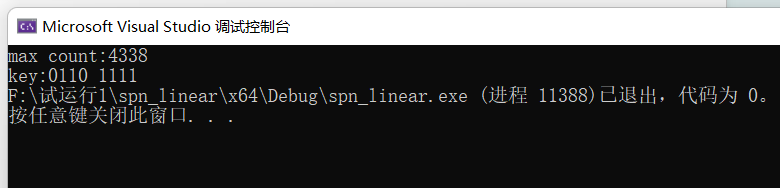
cout << " ";

for (int i = 0; i < 4; i++) {

cout << L2[i];

}

1. 实验结果：



推测正确密钥的count计数为：4338次

推测的SPN最后一轮2/4个s盒的密钥为：0110 1111

线性分析算法正确；