**于成俊 2112066**

**差分攻击研究报告**

1. **实验要求**

**要求实现差分攻击算法（P. 73）,分析出轮密钥**

1. **实验概述**

**差分密码分析是一个选择明文攻击。假设一个攻击者拥有大量的4元组（，，，）, 其中异或值 =是固定的。明文与用同一个密钥K加密分别得到密文与。对这些4元组中的每一个，将应用所有可能的候选密钥来对该密码的最后一轮进行解密。对每一个候选密钥，计算某些状态比特的值，并确定它们的异或是否有一个确定的值(即对给定输入异或值的最可能取的值)，如果是，就把对应于特定候选密钥的计数器加1。在这个过程的最后，我们希望具有最高频率的候选密钥含有真正密钥那些比特的取值。**

1. **实验原理**

**（自己的理解）**

**易知 = K K =，由此可以看出对于异或值，面对密钥可以直接“穿透过去”，所以选择异或值。但面对S盒就会发生改变。而S盒却有它的弱点，它可能分布不均匀，由此出现了差分分布表。从中选择概率最大的几个异或对，从而构成概率最大的差分链，再通过大量的明密文对和所有密钥来进行核对，找出成功率最高的密钥，即为所求。**

1. **实验步骤**
2. **首先实现SPN算法代码（因不是本次实验重点，所以代码详见cpp文件）**
3. **由P71页差分分布表可知，（1011，0010）概率最大，还可以看出（0000，0000）的概率为1。所以选择1011作为两个输入的异或，并让其他位全为0，而着重分析不等于0的部分。由此构造明文对，我让<2>=1011,<2>=0000,而其他位我通过用2进制数，每次循环+1来实现大量的明文对。代码如下：**

//明文

char\*\* x1 = new char\* [3000];

char\*\* x2 = new char\* [3000];

for (int i = 0; i <= 2999; i++) {

x1[i] = new char[17];

}

for (int i = 0; i <= 2999; i++) {

x2[i] = new char[17];

}

char A[5] = "0000";

char B[9] = "00000000";

bool a = false;

bool b = false;

for (int i = 0; i <= 2999; i++) {

//明文前四位

for (int k = 0; k <= 3; k++) {

if (a) {

if (A[k] == '0') {

A[k] = '1';

a = false;

}

else {

A[k] = '0';

}

}

x1[i][k] = x2[i][k] = A[k];

}

//明文后八位

for (int k = 0; k <=7; k++) {

if (b) {

if (B[k] == '0') {

B[k] = '1';

b = false;

}

else {

B[k] = '0';

}

}

x1[i][8 + k] = x2[i][8 + k] = B[k];

}

//明文的5-8位

x1[i][5] = x2[i][5] = '0';

x1[i][4] = x1[i][6] = x1[i][7] = '1';

x2[i][4] = x2[i][6] = x2[i][7] = '0';

x1[i][16] = x2[i][16] = '\0';

if ((i + 1) % 256 == 0) {

a = true;

}

b = true;

}

1. **用SPN算法生成密文**

//密文

char\*\* y1 = new char\* [3000];

char\*\* y2 = new char\* [3000];

for (int i = 0; i <= 2999; i++) {

y1[i] = new char[17];

y1[i] = SPN(x1[i], 4, 4, 4);

}

for (int i = 0; i <= 2999; i++) {

y2[i] = new char[17];

y2[i] = SPN(x2[i], 4, 4, 4);

}

1. **再通过二进制数来实现所有的密钥（256个）**

//密钥

char\*\* k = new char\* [256];

int count[256];

for (int i = 0; i <= 255; i++) {

k[i] = new char[9];

count[i] = 0;

}

for (int i = 0; i <= 7; i++) {

k[0][i] = '0';

}

k[0][8] = '\0';

for (int i = 1; i <= 255; i++) {

for (int j = 0; j <= 7; j++) {

if (k[i-1][j] == '0') {

k[i][j] = '1';

k[i][8] = '\0';

for (j++;;j++) {

if (j > 7) {

break;

}

k[i][j] = k[i - 1][j];

}

break;

}

else {

k[i][j] = '0';

}

}

k[i][8] = '\0';

}

1. **接着按书中的伪代码，写循环部分的代码**

for (int i = 0; i <= 2999; i++) {

bool t = true;

for (int j = 0,m=8; j <= 3; j++,m++) {

if (y1[i][j] == y2[i][j] && y1[i][m] == y2[i][m]) {}

else {

t = false;

break;

}

}

if (t) {

for (int j = 0; j <= 255; j++) {

char v2[2][5];

char v4[2][5];

char u2[3][5];

char u4[3][5];

for (int m = 4, n = 12, h = 0; h <= 3; m++, n++, h++) {

if (y1[i][m] == k[j][h]) {

v2[0][h] = '0';

}

else {

v2[0][h] = '1';

}

if (y1[i][n] == k[j][h+4]) {

v4[0][h] = '0';

}

else {

v4[0][h] = '1';

}

}

v2[0][4] = '\0';

v4[0][4] = '\0';

NS(v2[0], 4);

NS(v4[0], 4);

for (int n = 0; n <= 3; n++) {

u2[0][n] = v2[0][n];

u4[0][n] = v4[0][n];

}

u2[0][4] = '\0';

u4[0][4] = '\0';

////////////////////////

for (int m = 4, n = 12, h = 0; h <= 3; m++, n++, h++) {

if (y2[i][m] == k[j][h]) {

v2[1][h] = '0';

}

else {

v2[1][h] = '1';

}

if (y2[i][n] == k[j][h + 4]) {

v4[1][h] = '0';

}

else {

v4[1][h] = '1';

}

}

v2[1][4] = '\0';

v4[1][4] = '\0';

NS(v2[1], 4);

NS(v4[1], 4);

for (int n = 0; n <= 3; n++) {

u2[1][n] = v2[1][n];

u4[1][n] = v4[1][n];

}

u2[1][4] = '\0';

u4[1][4] = '\0';

for (int n = 0; n <= 3; n++) {

if (u2[0][n] == u2[1][n]) {

u2[2][n] = '0';

}

else {

u2[2][n] = '1';

}

if (u4[0][n] == u4[1][n]) {

u4[2][n] = '0';

}

else {

u4[2][n] = '1';

}

}

u2[2][4] = '\0';

u4[2][4] = '\0';

if (strcmp(u2[2], "0110") == 0 && strcmp(u4[2], "0110") == 0) {

count[j]++;

}

}

}

else {

continue;

}

}

int max = -1;

int index=0;

for (int i = 0; i <= 255; i++) {

if (count[i] > max) {

max = count[i];

index = i;

}

}

for (int i = 0; i <= 7; i++) {

cout << k[index][i];

if (i == 3) {

cout << " ";

}

}

**最终得到轮密钥的5-8位和13-16位。**