ZUC 算法 DDT 与 LAT 的实现

学　　院：软件与微电子学院

专　　业：计算机技术

学生姓名：吴宏凯

学　　号：1901210530

日　　期：2019-11-5

一、设计思路

1、 差分分布表 DDT：

1）基本思想：分析加密算法中输入的差异是否会带来输出差异的分布概率不平均。

2）符号约定：

m1：输入明文 1，m2：输入明文 2

c1：输出密文 1，c2：输出密文 2

difm：差分输入

difc：差分输出

S0：ZUC 使用的 s-box0

S1：ZUC 使用的 s-box1

ddt[256][256]：输出的差分分布表，统计差分输入输出对的出现次数

3）以 Naïve SPN 为例说明分布表的计算步骤：

已知 Naïve SPN 的 S-box 为：

input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

① 选定一个差分输入值 1011，人为构造一个输入明文 m1=0000，则对应的另一个输入明文 m2=1011⊕0000=1011；

② m1 和 m2 分别输入到 S0 中，根据上表可得到输出密文 c1=E=1110，c2=C=1100，则差分输出 difc=c1⊕c2=1110⊕1100=0010；

③ 根据①②的计算可得，在当前差分输入下，构造了第 1 个输入明文后，差分输出 0010 出现 1 次，于是在分布表 ddt[1011][0010]表项对应的值上加 1，表示计数器加 1。

④ 每选定一个差分输入，就需要构造 m1 从 0000-1111，重复①-③步，对分布表的对应表项不断累加。差分输入也需要从 0000-1111 遍历，才能构造完整的 DDT。

4）具体设计与实现：

①考虑到 ZUC 算法使用的 sbox 规模较大，且差分分析过程中不会对原有明文，差分输入的值做修改，所以只定义了一个 unsigned short 数组 m，数组长度为 256，按顺序存放 0x00-0xff，可重用为输入明文和差分输入，既可节约内存空间，又可在需要时直接查表使用而不需要实时生成。

②函数的主要部分，m1，difm 直接从数组 m 中获取,m2 为两者的异或，Sbox 使用 ZUC 给定的长度同为 256 的数组，m1、m2 作为下标，查表直接得到输出的密文 c1，c2，则差分输出为输出密文的异或。这样的基本操作重复 256*256 次。实现代码如下：

```
for(i=0;i<256;i++){
    for(j=0;j<256;j++){
        m2=m1[j]^difm[i];
        c1=S[m1[j]];
        c2=S[m2];
        difc=c1^c2;
        diftable[m1[i]][difc]+=1;
    }
```

```
}
```

③将生成的 DDT 输出成 TXT 文件，文件名为：“ZUC-DDT-S0.txt”，“ZUC-DDT-S1.txt”

```
FILE *fp;
fp=fopen("F:\\ZUC-DDT-S0.txt","w");
for(int i=0;i<256;i++){
    for(int j=0;j<256;j++){
        fprintf(fp,"%d,", diftable[i][j]);
    }
    fprintf(fp,"\n");
}
```

2、 线性近似表 LAT：

1）基本思想：分析加密算法中输入输出存在线性关系。

2）符号约定：

m1：输入明文 1，m2：输入明文 2，Xi 表示输入的第 i 位

c1：输出密文 1，c2：输出密文 2，Yj 表示输出的第 j 位

a：输入的线性表达式系数

b：输出的线性表达式系数

m_a=a1X1,a2X2,a3X3,a4X4：输入 m 与系数 a 相乘

c_b=b1Y1,b2Y2,b3Y3,b4Y4：输出 c 与系数 b 相乘

m_a_xor=$a1X1 \oplus a2X2 \oplus a3X3 \oplus a4X4$ ：最终的输入线性表达式

c_b_xor=$b1Y1 \oplus b2Y2 \oplus b3Y3 \oplus b4Y4$ ：最终的输出线性表达式

S0：ZUC 使用的 s-box0

S1：ZUC 使用的 s-box1

lat[256][256]：输出的线性近似表，统计当前参数对应的线性表达式成立的次数

3）以 Naïve SPN 为例说明分布表的计算步骤：

已知 Naïve SPN 的 S-box 为：

input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

① 选定一个输入表达式的系数 a1=0,a2=1,a3=1,a4=0，构造明文 m1=X1X2X3X4，则此时的输入线性表达式为 X2⊕X3；

② 将明文输入到 sbox 中得到相应的输出 Y1Y2Y3Y4，再选定一个输出表达式的系数 b1=1,b2=0,b3=1,b4=1，则得到 Y1⊕Y3⊕Y4；

③ 分别计算出①②中 X2⊕X3 与 Y1⊕Y3⊕Y4 的值，若相等，则在线性表的 lat[0110][1011]表项的值上加 1。

④ 每选定一组 a 系数，就需要构造 m1 从 0000-1111，与系数相乘，重复①-③步，计算出对应的 c1，再选定一组 b 系数与 c1 相乘，比较两表达式的值，相等则对分布表的对应表项不断累加。因此每个 a 系数下，b 系数需要从 0000 遍历到 1111，每组 a,b 系数下 m1 也需要从 0000-1111 遍历，而 a 系数本身也需要从 0000 遍历到 1111。

4）具体设计与实现：

①与 DDT 的设计相同，LAT 的实现中只定义了一个 unsigned short 数组 m，数组长度为 256，按顺序存放 0x00-0xff，可重用为输入明文和输入系数 a，输出系数 b。
②在生成 LAT 之前，编写了两个工具函数 valueAtBit()和 xorbit()：

```
//用于获取一个数的第 bit 位，索引从 1 开始
unsigned short valueAtBit(unsigned short num,int  bit) {
    return (num >> (bit-1)) & 1;
}
//用于对一个数做逐位两两异或
unsigned  short xorbit(unsigned short data){
    int i,j;
    unsigned short res;
    res=valueAtBit(data,16);
    for(j=15;j>0;j--){
        res=res^valueAtBit(data,j);
    }
    return res;
}
```

③因为输入明文 m1:X1X2X3X4 需要与输入系数 a:a1a2a3a4 按位相乘，编程实现中直接计算明文与系数按位与：m_a=m1&a；得到的 m_a 再使用②中的异或函数，得到 m_a_xor。
例如：a=0110,m=1011,按理论算法，则 a 与 m 按位相乘得表达式 X2⊕X3=1；按上述算法，a&m=0010=m_a，再对 m_a 逐位两两异或得到 m_a_xor=1，结果相同。

每选取一个系数 a，有 256 个对应的 b 系数，每组 a,b 系数，需要遍历 256 个 m，所以使用三个 for 循环嵌套：

```
    for(i=0;i<256;i++){
        //每个 a 有 256 个 b
        for(j=0;j<256;j++){
            //每个 a,b 有 256 个 m,c
            for(k=0;k<256;k++){
                m_a=a[i]&m1[k];
                m_a_xor=xorbit(m_a);
                c_b=b[j]&S[m1[k]];//此时 m 对应的输出
                c_b_xor=xorbit(c_b);
                if(m_a_xor==c_b_xor){
                    lintable[a[i]][b[j]]++;
                }
            }
        lintable[a[i]][b[j]]=lintable[a[i]][b[j]]-128;
        }
    }
```

⑤ 将生成的 LAT 输出成 TXT 文件，文件名为"ZUC-LAT-S0.txt","ZUC-LAT-S1.txt":

```
    fp=fopen("F:\\ZUC-LAT-S0.txt","w");
    for(int i=0;i<256;i++){
```

```
        for(int j=0;j<256;j++){
            fprintf(fp,"%d,", lintable[i][j]);
        }
        fprintf(fp,"\n");
    }
```

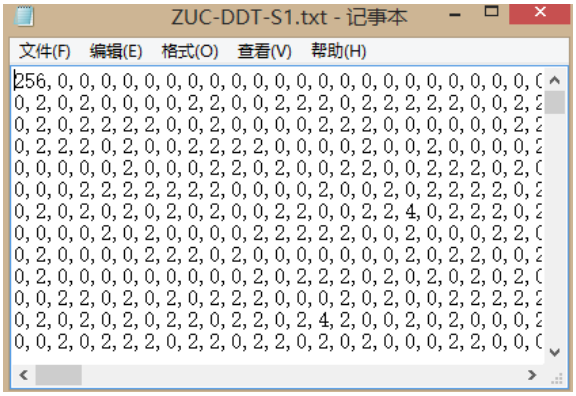
二、源代码

代码见附件

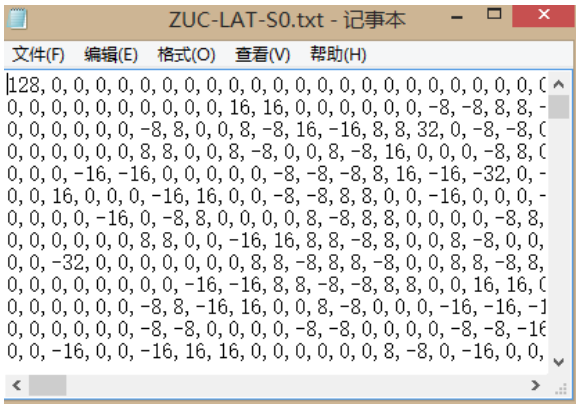
三、实验结果

生成四个 txt 文件，分别为
ZUC-DDT-S0.txt：存放 ZUC 中 S0 的 DDT;

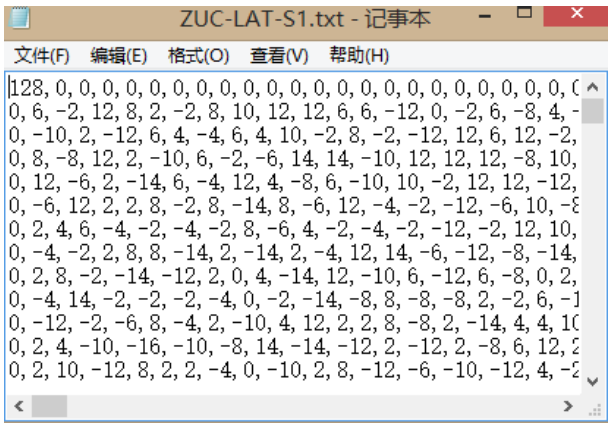
ZUC-DDT-S1.txt：存放 ZUC 中 S1 的 DDT;



ZUC-LAT-S0.txt：存放 ZUC 中 S0 的 LAT;



ZUC-LAT-S1.txt：存放 ZUC 中 S1 的 LAT;



文件详情见附件。