密码分析学 #2

XXX：202XX80XXXXXXXXX

2023 年 4 月 15 日

SPN 结构的差分密码分析

S 盒的差分分布表

该密码算法 S 盒的差分分布表如下：

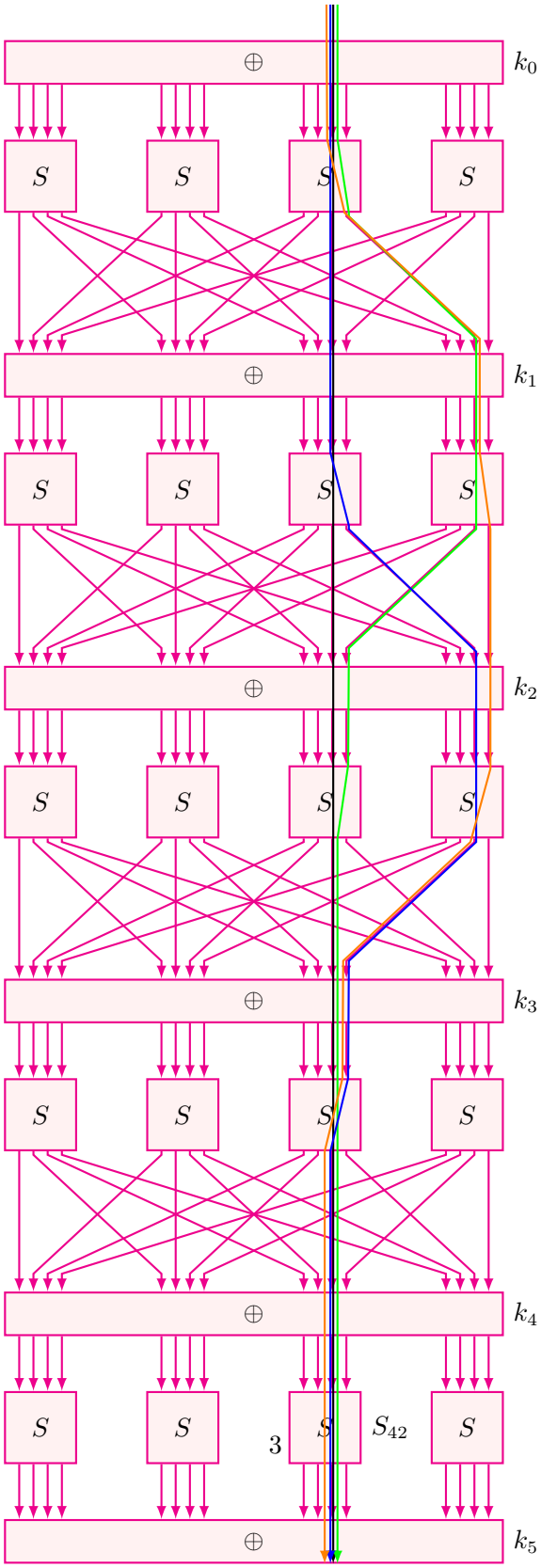
DDT	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	6	0	0	0	0	2	0	2	0	0	2	0	4	0
2	0	6	6	0	0	0	0	0	0	2	2	0	0	0	0	0
3	0	0	0	6	0	2	0	0	2	0	0	0	4	0	2	0
4	0	0	0	2	0	2	4	0	0	2	2	2	0	0	2	0
5	0	2	2	0	4	0	0	4	2	0	0	2	0	0	0	0
6	0	0	2	0	4	0	0	2	2	0	2	2	2	0	0	0
7	0	0	0	0	0	4	4	0	2	2	2	2	0	0	0	0
8	0	0	0	0	0	2	0	2	4	0	0	4	0	2	0	2
9	0	2	0	0	0	2	2	2	0	4	2	0	0	0	0	2
10	0	0	0	0	2	2	0	0	0	4	4	0	2	2	0	0
11	0	0	0	2	2	0	2	2	2	0	0	4	0	0	2	0
12	0	4	0	2	0	2	0	0	2	0	0	0	0	0	6	0
13	0	0	0	0	0	0	2	2	0	0	0	0	6	2	0	4
14	0	2	0	4	2	0	0	0	0	0	2	0	0	0	0	6
15	0	0	0	0	2	0	2	0	0	0	0	0	0	10	0	2

差分攻击的攻击方法

首先针对 S 盒产生差分分布表，根据差分分布表，寻找到出现概率较高的一个差分。然后产生足够多的差分明文对，生成相应的密文对。然后穷举密钥，将每一个密钥与密文对进行运算从而反向推导出 S 盒的输入差分，观察输入差分与我们所设想的差分的是否相同。若相同，则为当前穷举的密钥计一票，若不同，则不计票。其中最重要的步骤是得先找到那条高概率的差分。

高概率差分

根据《*The Block Cipher Companion*》书中所描绘的，对于该密码算法，有 4 条高概率的差分路径满足 $(0,0,2,0) \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} (0,0,2,0) \cdots\cdots (1)$


k_0
S
S
S
S
k_1
S
S
S
S
k_2
S
S
S
S
k_3
S
S
S
S
k_4
S
S
3
S
S_{42}
S
k_5

分别是：

图中黑色差分路径：

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

图中橙色差分路径：

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,0,2) \xrightarrow{\mathscr{R}} (0,0,0,1) \xrightarrow{\mathscr{R}} (0,0,1,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

图中绿色差分路径：

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,0,2) \xrightarrow{\mathscr{R}} (0,0,1,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

图中蓝色差分路径：

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,0,2) \xrightarrow{\mathscr{R}} (0,0,1,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

则对于差分 (1)，其出现的概率大于上述四条差分路径出现概率之和 $4 \times (\frac{6}{16})^4 = \frac{81}{1024}$。差分 (1) 出现的概率高于一个差分随机出现的概率 ($\frac{1}{16}$)，证明该差分路径将不会随机出现的差分受到干扰。

过滤

所谓过滤，就是将不属于该差分的明密文差分对剔除出去，在本次差分攻击中，可以看到，在图中的 S 盒 S_{42} 的输出差分为 2，那么根据差分分布表，S_{42} 的输出差分只可能是 $1, 2, 9, a$，据此就可以将密文输出差分不是 $1, 2, 9, a$ 的明文密文差分对排除。

计票

在这次攻击中，我们猜测密钥 K_5 的第 8-11 比特。对于这四比特的所有 16 种取值，将过滤之后得到的密文对中相应的比特位与密钥的这四比特进行异或，得到 S 盒的输出，反向推导出 S 盒的输入，进而计算它们的差分，若差分为 2，则为当前的密钥四比特计上一票。最终取投票最多的密钥为正确的密钥。

运行结果说明

安装并配置好 ruby 环境，在命令行模式下进入 diff-attack.rb 所在的目录，执行命令

```
$ruby diff-attack.rb
```

程序将产生一个随机密钥，并打印出来，同时提示我们这次攻击要猜测的密钥值（也就是打印出来的密钥的倒数 4-8 位，即最后一轮的轮密钥的第 8-11 位（以 0 为第一个索引））。之后稍等一段时间，程序将进行攻击。攻击结束后，程序将打印出票数最多的十个密钥值。格式为 {*the key value in decimal*} => *the ratio that the counter number to* 65536。

接着，程序将提示，攻击结束后猜测的密钥值为多少。若猜测正确，则打印出 *Attack successed*!。一个可能的程序输出如下：

```
the generated 96-bit random key is
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
     1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
    0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0]
the key we want to recover is subkey 5 bits 8-11: [1, 1, 0, 1]

Here are the sorted best 10 results from analysis:
{13} => 0.083526611328125
{15} => 0.05609130859375
{12} => 0.05328369140625
{3} => 0.0462646484375
{1} => 0.0462646484375
{14} => 0.0430908203125
{2} => 0.041839599609375
{0} => 0.041168212890625
{4} => 0.032440185546875
{7} => 0.026336669921875
Result:the most probable subkey 5 bits 8-11 is 13 in decimal whose binary is
     [1, 1, 0, 1]
Attack successed!
```

result