现代密码学

Modern Cryptography

张方国

中山大学计算机学院

Office: Room 305, IM School Building

E-mail: isszhfg@mail.sysu.edu.cn

HomePage: https://cse.sysu.edu.cn/content/2460

第七讲 分组密码（二）

- 线性密码分析（续）
- 差分密码分析

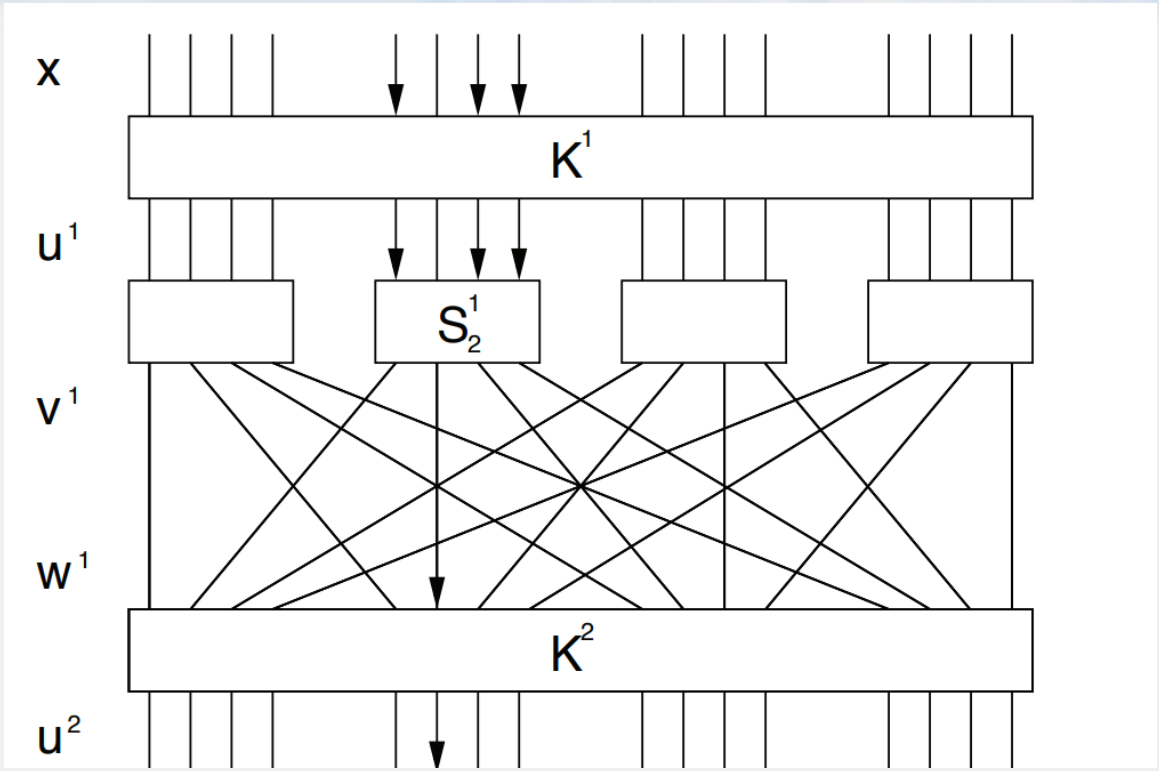
线性密码分析步骤

- 求找出一组S盒的线性逼近
- 导出整个SPN（除最后一轮）的线性逼近
- 利用已有的明密文对，测试候选密钥
- 输出密钥

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1

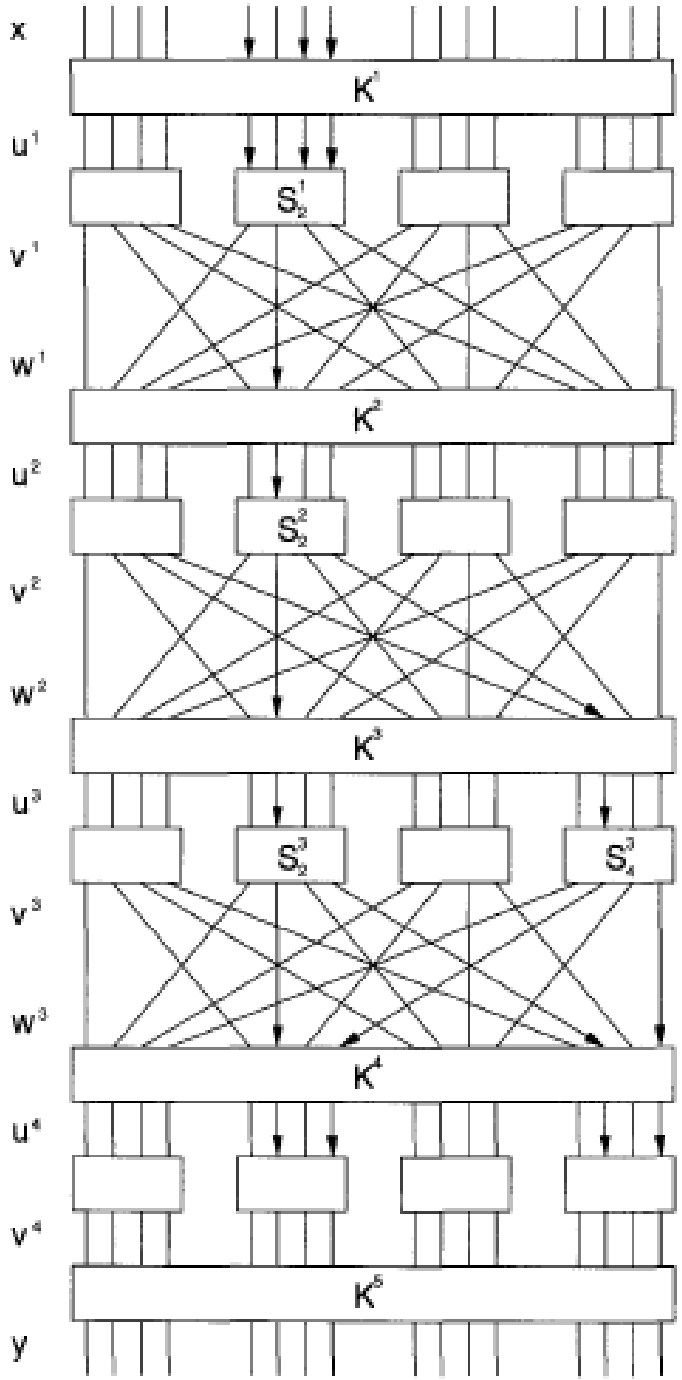
计算变量X=X1+X3+X4+Y2的偏差(3/4-1/2)

X1	X3	X4	Y2	X
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



In S_2^1, the random variable $\mathbf{T_1} = \mathbf{U_5^1} \oplus \mathbf{U_7^1} \oplus \mathbf{U_8^1} \oplus \mathbf{V_6^1}$ has bias $1/4$

- 测试K^2的第6比特



- In S_2^1, the random variable $\mathbf{T_1} = \mathbf{U_5^1} \oplus \mathbf{U_7^1} \oplus \mathbf{U_8^1} \oplus \mathbf{V_6^1}$ has bias $1/4$
- In S_2^2, the random variable $\mathbf{T_2} = \mathbf{U_6^2} \oplus \mathbf{V_6^2} \oplus \mathbf{V_8^2}$ has bias $-1/4$
- In S_2^3, the random variable $\mathbf{T_3} = \mathbf{U_6^3} \oplus \mathbf{V_6^3} \oplus \mathbf{V_8^3}$ has bias $-1/4$
- In S_4^3, the random variable $\mathbf{T_4} = \mathbf{U_{14}^3} \oplus \mathbf{V_{14}^3} \oplus \mathbf{V_{16}^3}$ has bias $-1/4$

$$\mathbf{T_1} \oplus \mathbf{T_2} \oplus \mathbf{T_3} \oplus \mathbf{T_4}$$

has bias equal to $2^3(1/4)(-1/4)^3 = -1/32$.

$$\begin{aligned}
\mathbf{T_1} &= \mathbf{U_5^1} \oplus \mathbf{U_7^1} \oplus \mathbf{U_8^1} \oplus \mathbf{V_6^1} = \mathbf{X_5} \oplus \mathbf{K_5^1} \oplus \mathbf{X_7} \oplus \mathbf{K_7^1} \oplus \mathbf{X_8} \oplus \mathbf{K_8^1} \oplus \mathbf{V_6^1} \\
\mathbf{T_2} &= \mathbf{U_6^2} \oplus \mathbf{V_6^2} \oplus \mathbf{V_8^2} \qquad = \mathbf{V_6^1} \oplus \mathbf{K_6^2} \oplus \mathbf{V_6^2} \oplus \mathbf{V_8^2} \\
\mathbf{T_3} &= \mathbf{U_6^3} \oplus \mathbf{V_6^3} \oplus \mathbf{V_8^3} \qquad = \mathbf{V_6^2} \oplus \mathbf{K_6^3} \oplus \mathbf{V_6^3} \oplus \mathbf{V_8^3} \\
\mathbf{T_4} &= \mathbf{U_{14}^3} \oplus \mathbf{V_{14}^3} \oplus \mathbf{V_{16}^3} \qquad = \mathbf{V_8^2} \oplus \mathbf{K_{14}^3} \oplus \mathbf{V_{14}^3} \oplus \mathbf{V_{16}^3}.
\end{aligned}$$

$$\begin{aligned}
&\mathbf{X_5} \oplus \mathbf{X_7} \oplus \mathbf{X_8} \oplus \mathbf{V_6^3} \oplus \mathbf{V_8^3} \oplus \mathbf{V_{14}^3} \oplus \mathbf{V_{16}^3} \\
&\qquad \oplus \mathbf{K_5^1} \oplus \mathbf{K_7^1} \oplus \mathbf{K_8^1} \oplus \mathbf{K_6^2} \oplus \mathbf{K_6^3} \oplus \mathbf{K_{14}^3} \qquad (3.1)
\end{aligned}$$

$$\begin{aligned}
\mathbf{V_6^3} &= \mathbf{U_6^4} \oplus \mathbf{K_6^4} \\
\mathbf{V_8^3} &= \mathbf{U_{14}^4} \oplus \mathbf{K_{14}^4} \\
\mathbf{V_{14}^3} &= \mathbf{U_8^4} \oplus \mathbf{K_8^4} \\
\mathbf{V_{16}^3} &= \mathbf{U_{16}^4} \oplus \mathbf{K_{16}^4}
\end{aligned}$$

$$\begin{aligned}
&\mathbf{X_5} \oplus \mathbf{X_7} \oplus \mathbf{X_8} \oplus \mathbf{U_6^4} \oplus \mathbf{U_8^4} \oplus \mathbf{U_{14}^4} \oplus \mathbf{U_{16}^4} \\
&\quad \oplus \mathbf{K_5^1} \oplus \mathbf{K_7^1} \oplus \mathbf{K_8^1} \oplus \mathbf{K_6^2} \oplus \mathbf{K_6^3} \oplus \mathbf{K_{14}^3} \oplus \mathbf{K_6^4} \oplus \mathbf{K_8^4} \oplus \mathbf{K_{14}^4} \oplus \mathbf{K_{16}^4} \qquad (3.2)
\end{aligned}$$

$$\mathbf{X_5} \oplus \mathbf{X_7} \oplus \mathbf{X_8} \oplus \mathbf{U_6^4} \oplus \mathbf{U_8^4} \oplus \mathbf{U_{14}^4} \oplus \mathbf{U_{16}^4} \qquad (3.3)$$

线性密码分析要求找出一组S盒的线性逼近，这组线性逼近能够用来导出一个整个SPN（除最后一轮）的线性逼近。

通过图3.3的分析，我们得到

$$\mathbf{T}_1 \oplus \mathbf{T}_2 \oplus \mathbf{T}_3 \oplus \mathbf{T}_4 = \mathbf{X}_5 \oplus \mathbf{X}_7 \oplus \mathbf{X}_8 \oplus \mathbf{U}_6^4 \oplus \mathbf{U}_8^4 \oplus \mathbf{U}_{14}^4 \oplus \mathbf{U}_{16}^4$$

$$\oplus \mathbf{K}_5^1 \oplus \mathbf{K}_7^1 \oplus \mathbf{K}_8^1 \oplus \mathbf{K}_6^2 \oplus \mathbf{K}_6^3 \oplus \mathbf{K}_{14}^3 \oplus \mathbf{K}_6^4 \oplus \mathbf{K}_8^4 \oplus \mathbf{K}_{14}^4 \oplus \mathbf{K}_{16}^4 \qquad (3.2)$$

因为密钥比特固定，所以事实上

$$\mathbf{K}_5^1 \oplus \mathbf{K}_7^1 \oplus \mathbf{K}_8^1 \oplus \mathbf{K}_6^2 \oplus \mathbf{K}_6^3 \oplus \mathbf{K}_{14}^3 \oplus \mathbf{K}_6^4 \oplus \mathbf{K}_8^4 \oplus \mathbf{K}_{14}^4 \oplus \mathbf{K}_{16}^4$$

具有固定的值。

因此我们可以使用随机变量

$$\mathbf{X}_5 \oplus \mathbf{X}_7 \oplus \mathbf{X}_8 \oplus \mathbf{U}_6^4 \oplus \mathbf{U}_8^4 \oplus \mathbf{U}_{14}^4 \oplus \mathbf{U}_{16}^4$$

具有偏离0的偏差这一事实允许我们进行线性密码攻击。

假设我们拥有同一未知密钥K加密的T对明-密文。用T来表示T对明-密文的集合。线性攻击将使我们获得$K_{<2>}^5$和$K_{<4>}^5$的8比特密钥，即

$$K_5^5, K_6^5, K_7^5, K_8^5, K_{13}^5, K_{14}^5, K_{15}^5, K_{16}^5$$

这些正是与S盒S_2^4和S_4^4的输出相异或的8比特密钥。256种可能，每一种可能都叫做一个候选子密钥。

对每一个$(x,y)\in$T及每一个子密钥，计算y的一个部分解密并得到$u^4_{<2>}$和$u^4_{<4>}$，然后计算

$$x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16} \quad (3.4)$$

之值。保持对应于这256个候选子密钥的256个计数器，每当式(3.4)取值为0时，就将对应于该子密钥的计数器加1。

在计数过程中，我们希望大多数的计数器接近于T/2，而真正的候选子密钥对应的计数器具有接近于$T/2\pm T/32$之值，这有助于我们确定正确的8个子密钥比特。

线性密码分析算法

Algorithm 3.2: LINEARATTACK($\mathcal{T}, T, {\pi_S}^{-1}$)

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y) \in \mathcal{T}$
  do
    for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
      do
        $v^4_{<2>} \leftarrow L_1 \oplus y_{<2>}$
        $v^4_{<4>} \leftarrow L_2 \oplus y_{<4>}$
        $u^4_{<2>} \leftarrow {\pi_S}^{-1}(v^4_{<2>})$
        $u^4_{<4>} \leftarrow {\pi_S}^{-1}(v^4_{<4>})$
        $z \leftarrow x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16}$
        if $z = 0$
          then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do
    $Count[L_1, L_2] \leftarrow |Count[L_1, L_2] - T/2|$
    if $Count[L_1, L_2] > max$
      then
        $max \leftarrow Count[L_1, L_2]$
        $maxkey \leftarrow (L_1, L_2)$
output $(maxkey)$

算法分析

算法3.2给出了这个特殊的线性攻击算法，输出的maxkey包含了该攻击确定出的具有最大可能的8个子密钥比特。

算法3.2的简单分析。

一般来说，一个基于偏差为ε的线性逼近的线性攻击要想获得成功，所需要的明-密文对数目T要接近于$c\varepsilon^{-2}$，对某个小的常数c。算法3.2取$T = 8000$。

差分分析

- 1990年国际密码年会上， Eli Biham和Adi Shamir首次提出了对DES的差分攻击，从此国际上开启了分组密码分析的热潮。
- 更早是由IBM在1974年提出，不过IBM将差分密码分析设为机密。
- 差分分析方法是一种选择明文攻击，其基本思想时通过分析特定明文差分对结果密文差分的影响来获得可能性最大的密钥。它主要使用于攻击迭代密码体制。

差分分析

差分密码分析是一个选择明文攻击。它与线性密码分析有些相同，但它们的主要区别在于差分密码分析包含了将两个输入的异或与其相对应的两个输出的异或相比较。一般来说，我们将要考察两个二元串x和x^*，它们有固定的异或值$x' = x \oplus x^*$。

假设攻击者拥有大量的4元组(x, x^*, y, y^*)，其中异或值$x' = x \oplus x^*$是固定的。明文x和x^*用同一个密钥K加密分别得到密文y和y^*。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，如果是，就把对应于特定候选密钥的计数器加1。

我们希望最有最高频率的候选密钥含有真正密钥那些比特的取值。

差分分析

定义3.1 设$\pi_s : \{0,1\}^m \to \{0,1\}^n$是一个S盒。考虑长为$m$的有序比特串对$(x, x^*)$，我们称$S$盒的输入异或为$x \oplus x^*$，输出异或为$\pi_s(x) \oplus \pi_s(x^*)$。注意输出异或是一个$n$长比特串。对任何$x' \in \{0,1\}^m$，定义集合$\triangle(x')$为包含所有具有输入异或值$x'$的有序对$(x, x^*)$。

可见，集合$\triangle(x')$包含2^m对，并且

$$\triangle(x') = \{(x, x \oplus x') : x \in \{0,1\}^m\}$$

对集合$\triangle(x')$中的每一对，都能计算它们关于S盒的输出异或，然后我们能将所有输出异或的值列成一张结果分布表，一共有2^m个输出异或，它们的值分布在2^n个可能值之上。一个非均匀的输出分布将会是一哥成功差分攻击的基础。

差分分析

例3.3 设输入异或$x' = 1011$，则

$$\triangle(1011) = \{(0000,1011),(0001,1010),\ldots,(1111,0100)\}$$

对$\triangle(1011)$中的每一有序对，可计算出π_s的输出异或。

则y'的分布如下：

$$(0000,0),(0001,0),(0010,8),(0011,0)$$

$$(0100,0),(0101,2),(0110,0),(0111,2)$$

$$(1000,0),(1001,0),(1010,0),(1011,0)$$

$$(1100,0),(1101,2),(1110,0),(1111,2)$$

差分分析

例3.3 在16个可能的输出异或中实际5个出现，这个特殊的例子具有一个非常不均匀的分布。对于长为m的比特串x'和长为n的比特串y'，定义

$$N_D(x',y') = |\{(x,x^*) \in \triangle(x') : \pi_s(x) \oplus \pi_s(x^*) = y'\}|$$

$N_D(x',y')$记下了输入异或等于x'，输出异或等于y'的对数（对某一给定的S盒）。

根据此原理，我们能计算出所有可能的$N_D(a',b')$（图3.4）。

差分分析

a' \ b'	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	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

差分分析

例3.3 在例3.1中的SPN，其中第r轮第i个S盒的输入是$u^r_{<r>}$，并且$u^r_{<r>} = w^{r-1}_{<r>} \oplus K^r_{<i>}$，一个输入异或可以这样计算：

$$u^r_{<r>} \oplus (u^r_{<r>})^* = (w^{r-1}_{<r>} \oplus K^r_{<i>}) \oplus ((w^{r-1}_{<r>})^* \oplus K^r_{<i>}) = w^{r-1}_{<r>} \oplus (w^{r-1}_{<r>})^*$$

因此，该输入异或并不依赖于第r轮的子密钥，它仅等于第$r-1$轮置换的输出异或。

差分分析

设a'表示一个输入异或，b'表示一个输出异或，(a',b')对叫做一个差分。

差分分布表中的每一项均导致了一个异或扩散率（简称扩散率）。对应于差分(a',b')的扩散率$R_p(a',b')$如下定义：

$$R_p(a',b') = \frac{N_D(a',b')}{2^m}$$

$R_p(a',b')$也可被理解为一个条件概率：Pr[输出异或=b'—输入异或=a'a]=$R_p(a',b')$。

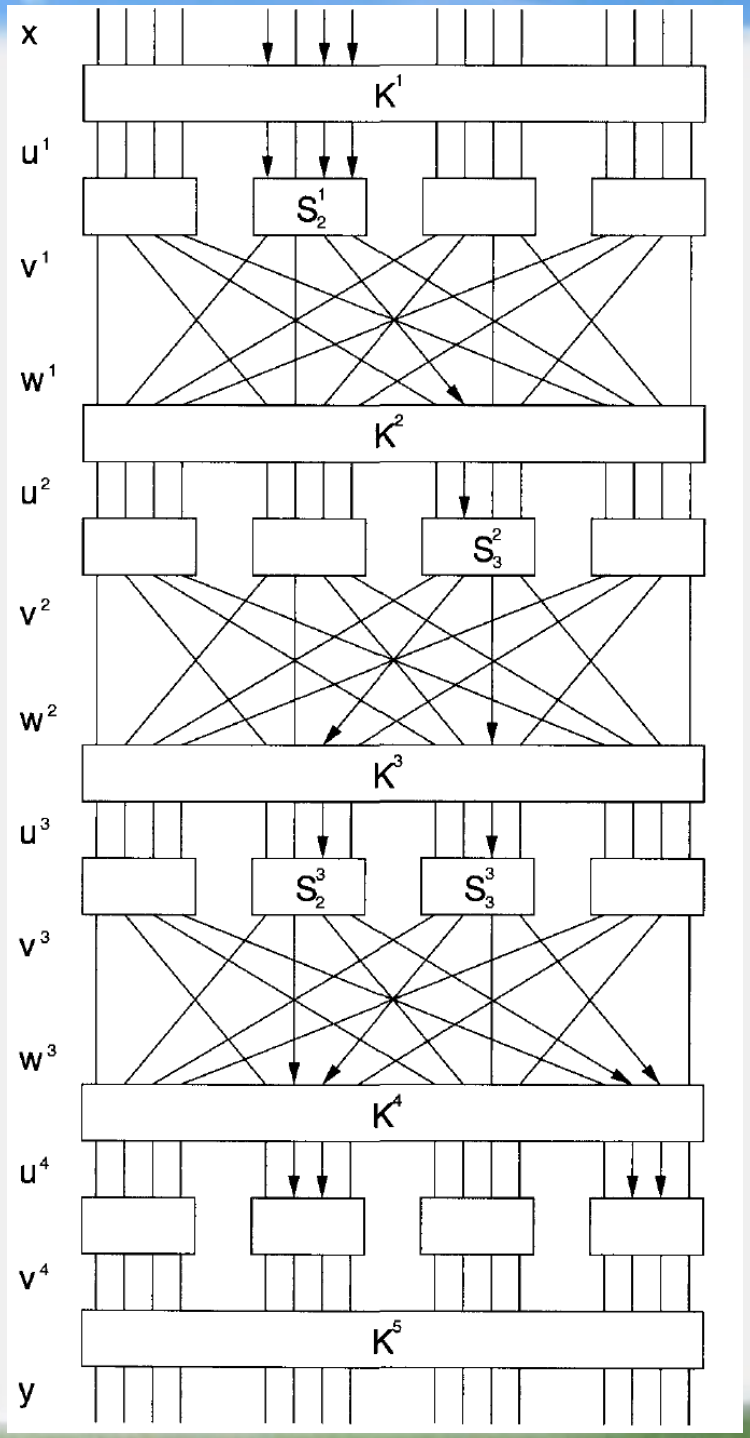
差分分析

（**差分链**）假设可在SPN的连续的轮中找到扩散率，使得任何一轮的一个差分输入异或实际上是前一轮差分的置换输出异或，这样，这些差分就构成了一个差分链。

假设差分链中不同的扩散率相互独立，由这个假设，就能把一个差分链里的扩散率相乘来获得整个差分链的扩散率。

例如图3.5。

差分链



Algorithm 3.3: DIFFERENTIALATTACK($\mathcal{T}, T, {\pi_S}^{-1}$)

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y, x^*, y^*) \in \mathcal{T}$
  do **if** $(y_{<1>} = (y_{<1>})^*)$ **and** $(y_{<3>} = (y_{<3>})^*)$
    then **for** $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
      do $v^4_{<2>} \leftarrow L_1 \oplus y_{<2>}$
        $v^4_{<4>} \leftarrow L_2 \oplus y_{<4>}$
        $u^4_{<2>} \leftarrow {\pi_S}^{-1}(v^4_{<2>})$
        $u^4_{<4>} \leftarrow {\pi_S}^{-1}(v^4_{<4>})$
        $(v^4_{<2>})^* \leftarrow L_1 \oplus (y_{<2>})^*$
        $(v^4_{<4>})^* \leftarrow L_2 \oplus (y_{<4>})^*$
        $(u^4_{<2>})^* \leftarrow {\pi_S}^{-1}((v^4_{<2>})^*)$
        $(u^4_{<4>})^* \leftarrow {\pi_S}^{-1}((v^4_{<4>})^*)$
        $(u^4_{<2>})' \leftarrow u^4_{<2>} \oplus (u^4_{<2>})^*$
        $(u^4_{<4>})' \leftarrow u^4_{<4>} \oplus (u^4_{<4>})^*$
        if $((u^4_{<2>})' = 0110)$ **and** $((u^4_{<4>})' = 0110)$
          then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do **if** $Count[L_1, L_2] > max$
    then $max \leftarrow Count[L_1, L_2]$
      $maxkey \leftarrow (L_1, L_2)$
output $(maxkey)$

算法3.3利用了一种叫做过滤的操作。使得差分成立的4元组(x,x^*,y,y^*)叫做一个正确对，正是这些正确对使得我们能够确定相关的密钥比特。

一个正确对应该满足

$$(u^4_{<1>})' = (u^4_{<3>})' = 0000$$

因此，一个正确对必须满足$y_{<1>} = (y_{<1>})^*$和$y_{<3>} = (y_{<3>})^*$。如果4元组不满足这个条件，那么它就不是一个正确对，必须丢弃。这种过滤操作能提高差分攻击的效率。

算法3.3的工作流程如下：对每个4元组(x,x^*,y,y^*)，首先完成过滤操作。

如果(x,x^*,y,y^*)是一个正确对，那么就测试每一个可能的候选子密钥(L_1,L_2)，并且如果观察到一个确定的异或值，就将对应于(L_1,L_2)的计数器加1.

当4元组(x,x^*,y,y^*)的数量T接近于$c\varepsilon^{-1}$时，一个基于扩散率为ε的差分链攻击一般会成功。

算法3.3能成功，如果T在50 100之间，因为$\varepsilon^{-1} = 38$。