利用差分密码分析确定 SPN 分组加密算法的(轮)密钥

学号: 2310764 姓名: 王亦辉 专业: 计科

1 原理介绍

差分密码分析是选择明文攻击，也就是说，我们只有 K 是不知道的。我们知道 S 盒、P 盒，加密轮数等一切与密码体制有关的信息，同时可以无限制使用加密机 (即使不知道 K，也能将明文加密成密文)。

在此基础上，差分密码分析通过利用

1. 两个输入比特串 x, x^* 的异或 $x \oplus x^*$ 的值不会受加密中 K^r 异或的影响这一特性，以及
2. 分组长度较长时，如 $l * m = 16$ ，异或相同的 (x, x^*) 组合非常多 (2^{16} 种) ，因此猜测时可以把它们当成离散随机变量，研究它们的异或经过一轮 S 盒、P 盒的分布，
   从概率论的角度来分析出最后一轮加密的影响，最后选择适当的差分链猜测密钥。

我们需要实现如下差分攻击算法对轮密钥进行猜测。
注意到，这里的伪代码是针对书中选取的差分 $x^{'} = 0000\_1011\_0000\_0000$ 进行书写的，为了猜测出最后一轮的密钥 $K^{Nr+1} = K^5$ 我们需要做一些改动。
此外，书中伪代码与我的实现都是基于 $l = m = Nr = 4$，密钥长度为 32 并采用 k_{4r-3} 进行每轮密钥选取这些前提上进行的。

算法 3.3　差分攻击$(\mathcal{T}, T, \pi_S^{-1})$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F,F)
  do $\text{Count}[L_1, L_2] \leftarrow 0$
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** $\text{Count}[L_1, L_2] \leftarrow \text{Count}[L_1, L_2] + 1$

$\max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ **to** (F,F)

do
- **if** $\text{Count}[L_1, L_2] > \max$
  - **then**
    - $\max \leftarrow \text{Count}[L_1, L_2]$
    - $\text{maxkey} \leftarrow (L_1, L_2)$

output(maxkey)

2　关键代码实现与密钥猜测

全部代码见 https://github.com/CraftOldWang/Cryption

以下代码是对上图中伪代码的实现：
其中，`T_set` 为差分为特定值 x' 的四元组 (x, x^*, y, y^*) 的迭代器；`times` 为尝试的有效对个数；`difx` 指定最后一轮 S 盒的两个输入的异或，即原伪代码的 `0110` 部分，`None` 代表差分为 `0000`。

流程为：
筛选掉必然不可能是正确对的四元组以提高效率，然后对正确对，检查候选子密钥并对满足条件的子密钥计数器累加 1，最后输出计数器最大的候选子密钥。

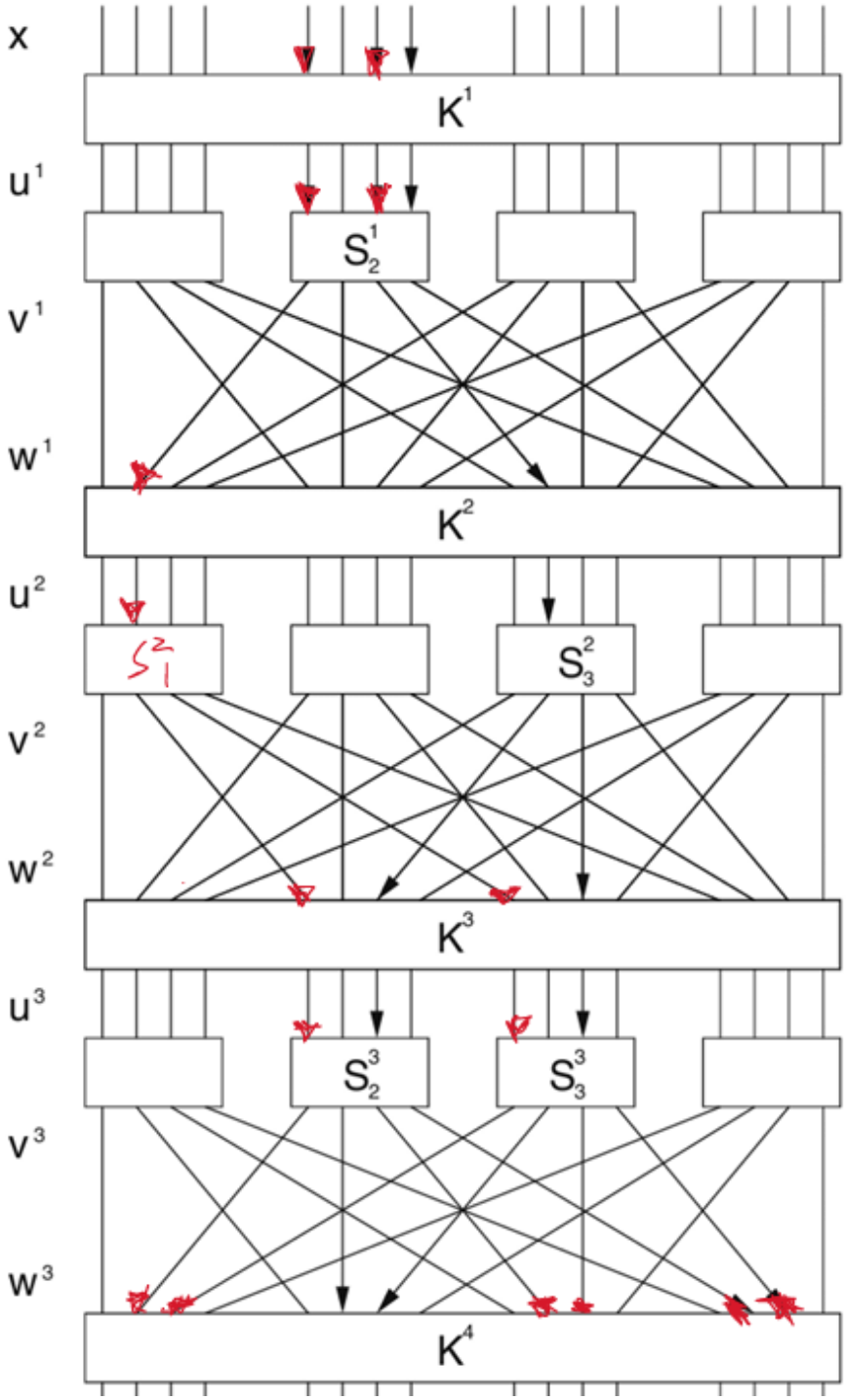
```
1 def differential_attack(T_set, times, dif1=None, dif2=None, dif3=None,
  dif4=None):
2     key_candidate = defaultdict(int)
```

```
    diffs = [dif1, dif2, dif3, dif4]
    active_indices = [i for i, d in enumerate(diffs) if d is not None]
    active_diffs = [diffs[i] for i in active_indices]

    cur_time = 0  # 有效次数
    for _, _, y, y_star in T_set:
        # 过滤无效对
        is_effective = True
        for i, dif in enumerate(diffs):
            if dif == None and y[4*i:4*i+4] != y_star[4*i:4*i+4]:
                is_effective = False
                break
        if not is_effective:
            continue

        y_block = [y[4*i:4*i+4] for i in active_indices]
        y_star_block = [y_star[4*i:4*i+4] for i in active_indices]
        for key in product(range(16), repeat=len(active_indices)):
            key_bin = tuple(f"{x:04b}" for x in key)
            if is_dif_satisfy(key_bin, y_block, y_star_block
,active_diffs):
                key_candidate[key_bin] += 1

        cur_time +=1
        if cur_time >= times:
            break

    maxkey = max(key_candidate, key= key_candidate.get)
    return maxkey
```

设最后一轮的轮密钥为 $K^5 = (L_1, L_2, L_3, L_4)$ 其中 L_i 长度为 4bit，那么通过 `differential_attack (T_set, times, dif1=None, dif2="0110", dif3=None, dif4="0110"):` 可以猜测出 L_2, L_4。

为了猜测出 L_1, L_3 还需要再选取一个差分 x'，使得 y' 的 1-4，9-12 位不为全 0。尝试后我选取了 `0000_1010_0000_0000` ，扩散过程如下图。查表可以计算

$$R_p(0000\_1010\_0000\_0000, 0110\_0000\_0110\_0110) = \left(\frac{6}{16}\right)^2 \times \left(\frac{4}{16}\right)^2 = \frac{9}{1024}$$



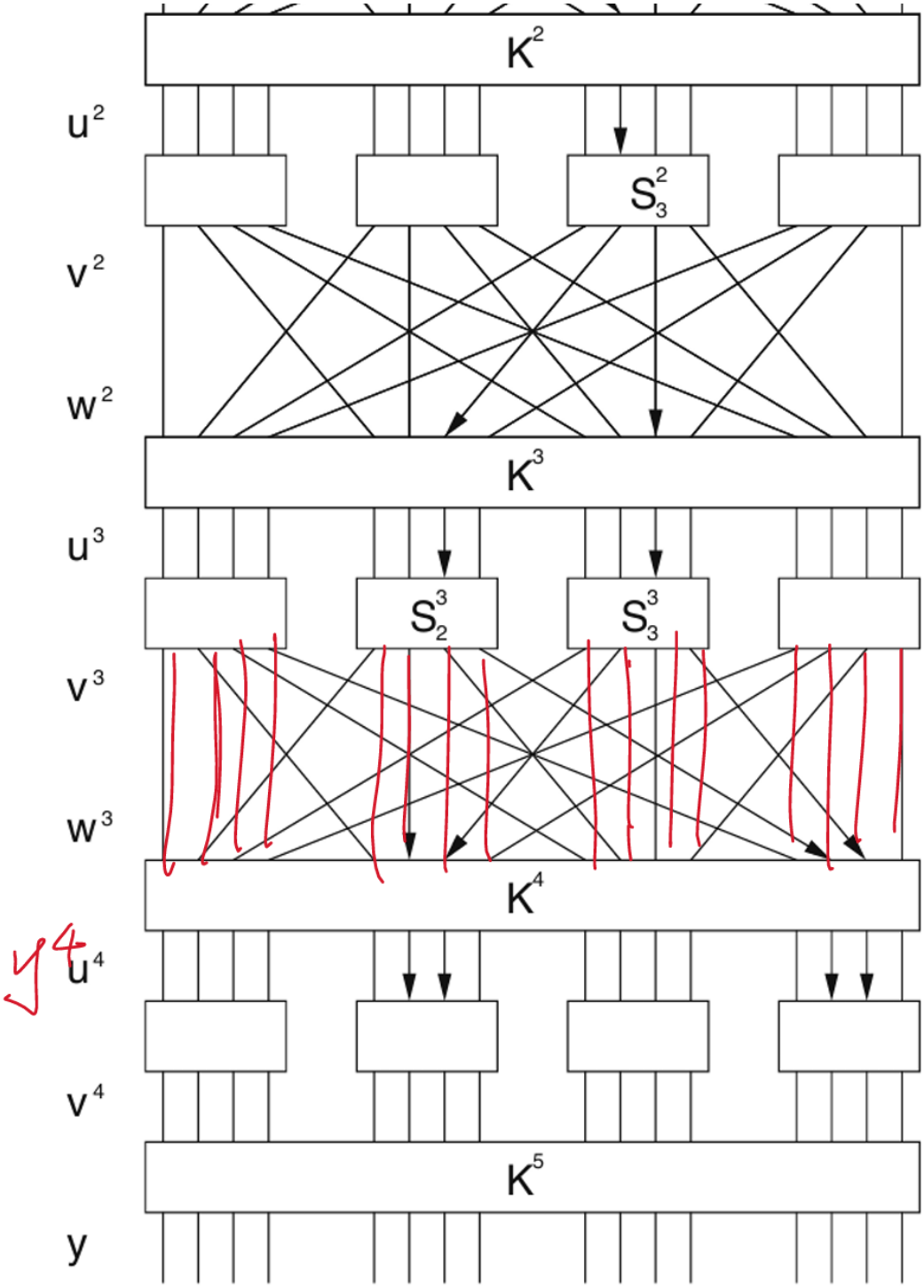
最终，我们可以猜测出最后一轮加密用的密钥 $K^5 = (L_1, L_2, L_3, L_4)$。即密钥 K 的 17-32 位。

通过随机实验，测试出当使用的有效对数量为 300 时，准确率为 92%，且观察发现全都错在 L_1, L_3 未猜对，这是由于我选取的 x' 的扩散率较低导致的。

3 剩余部分

如果我们需要拿到整个密钥 K，则需一轮轮往前推，由于已经获取了最后 16 位密钥，即 K^5 ，我们可以通过 π_p 的逆映射以及 π_s 的逆映射往前一层层倒推。

如下图，观察到加密过程中的相似性，由于我们已知 π_p，因此可以将上面部分看作 `Nr= 3` 的情形，先经过三轮加密，结果与 K^4 异或再输出。
由于我们已知 K^5 ，因此可以从选择明文攻击得到的 y 倒推回 y^4，从而化归为其他前提不变，`Nr` 改为 3，密钥长度变为 28 的情形。
针对 `Nr = 3` 的情形，可以实施类似的差分攻击，可以得到 K 的 13-16 位。依次类推，再三回，得到整个密钥 K。


K^2
u^2
S_3^2
v^2
w^2
K^3
u^3
S_2^3
S_3^3
v^3
w^3
K^4
u^4
v^4
K^5
y