

周报-向嘉豪 (2024-11-18)

Abstract: 本周主要对 OPO (Optimization of Permutation Operation) 算法进行了优化和重构。通过引入贪心递归策略解决了局部最优解问题，并重构了 PPO 类结构以提高代码可维护性。将优化后的 OPO 算法应用于 AES 的 ShiftRow 操作，在 STM32L475 平台上进行测试，相比 [SS16] 的实现，执行周期数从 8932 降至 8068，性能提升 9.7%，同时降低了 Flash 内存占用。

下周计划: 完成论文撰写并将重。

0.1 OPO 算法优化

此前实现的 Optimization of Permutation Operation (OPO) 图 1a, 存在求解, 遇见局部最优解的问题, 通过引入贪心递归策略, 来找到全局最优解, 优化后的 OPO 算法如图 1b所示。

```
Algorithm 1 Optimization of Permutation Operation (OPO)
Input:  $P$  is a set composed of pairs  $(p, m)$ , where  $p$  is the number of shifts, and  $m$  is the mask.
Output:  $P'$  is an optimized set composed of pairs  $(p, m)$ .
1:  $P' \leftarrow \emptyset$ 
2: while  $0 \neq \text{len}(P)$  do
3:    $(p, m) \leftarrow \text{MinShift}(P)$  {Find the minimum number of shifts}
4:    $P \leftarrow P - (p, m)$ 
5:   if  $(p, m)$  in  $P'$  then
6:      $(p, m_1) \leftarrow (p, m)$  in  $P'$ 
7:      $P' \leftarrow P' - (p, m_1)$ 
8:      $P' \leftarrow P' + (p, m_1 \vee m)$  {Proposition 1}
9:   else
10:    if  $\text{Split}((p, m), P')$  is not empty then
11:       $(p_1, m_1), (p_2, m) \leftarrow \text{Split}((p, m), P')$  {Proposition 2}
12:       $P \leftarrow P + (p_1, m_1)$ 
13:       $P \leftarrow P + (p_2, m)$ 
14:    else
15:       $P' \leftarrow P' + (p, m)$ 
16:    end if
17:  end if
18: end while
19: return  $P'$ 
```

(a) 原始 OPO 算法

```
Algorithm 1 Optimization of Permutation Operations (OPO)
Input: Set of pairs  $P(p, m)$ , optimization index  $n$ 
Output: Optimized set of pairs  $P'$ 
1:  $P' \leftarrow P$ 
2: if  $n = \text{Length}(P')$  then
3:   return  $P'$ 
4: end if
5:  $P'_1 \leftarrow \text{OPO}(P', n+1)$ 
6: if  $\text{Split}(P'[n]) \neq \emptyset$  then
7:    $(p_1, m_1), (p_2, m_2) \leftarrow \text{Split}(P'[n])$ 
8:    $P' \leftarrow P' \cup \{(p_1, m_1), (p_2, m_2)\}$ 
9:   Delete  $P'[n]$ 
10:   $P'_2 \leftarrow \text{OPO}(P', n)$ 
11: end if
12: return Better( $P'_1, P'_2$ )
```

(b) 优化后 OPO 算法

图 1: OPO 算法优化

在对图 1b实现时, 发现旧版存在, 需要手动计算中间寄存器与汇编转化的问题, 因此我们重构了, 使用 PPO 类来结构化组织 PPO 操作, 如图 2所示。从之前的 $[(3, [8, 11, 10, 2]), (4, [5, 10, 4, 6, 12, 15]), (5, [14, 9])]$ 表示的操作序列, 转化为 PPO 类的操作序列, $[('and', 'r1', 'r0', '0xaa000000'), ('ror', 'r1', 'r1', 0), ('or', 'r14', 'r14', 'r1')]$, 类汇编组织结构。

```
class PPO:
> def __init__(self, temp, origin, dest, mask, shift):...
>
> def __eq__(self, value: "PPO") -> bool:...
>
> def toList(self):...
>
> @staticmethod
> def merge(ppo_list):...
```

图 2: PPO 类结构

0.2 OPO 优化 AES 算法

在 [SS16] 中提出的 AES 算法的基础上, 我们使用 OPO 算法进行优化, 通过 OPO 算法优化后的 AES 算法的 ShiftRow。以下为其中未优化代码 Listing 1和优化后代码 Listing 2的对比。通过我们的优化, 可以看出其中的 PPO 操作序列减少了 2 次。

由于性能测试框架和测试芯片的不同, [SS16] 中给出的实现周期数为 3234, 而我们将其代码放入我们 LCB 测试框架 stm32l475 测试出来的数据为 8932。为公平比较, 此处我们将使用 LCB, 测试我们的优化代码, 从表 1中可以看出, 我们的优化代码的周期数为 8068, 提升了 9.7%。

Listing 1: 原始汇编代码实现

```

1 # 操作序列
2 r1 r0 r14 0xaa000000 0
3 r1 r0 r14 0x2a 2
4 r1 r0 r14 0xa00 4
5 r1 r0 r14 0x20000 6
6 r1 r0 r14 0x80 26
7 r1 r0 r14 0xa000 28
8 r1 r0 r14 0xa80000 30
9 # 汇编代码
10 uxtb.w r12, r9
11 ubfx r5, r9, #14, #2
12 eor r12, r12, r5, lsl #8
13 ubfx r5, r9, #8, #6
14 eor r12, r12, r5, lsl #10
15 ubfx r5, r9, #20, #4
16 eor r12, r12, r5, lsl #16
17 ubfx r5, r9, #16, #4
18 eor r12, r12, r5, lsl #20
19 ubfx r5, r9, #26, #6
20 eor r12, r12, r5, lsl #24
21 ubfx r5, r9, #24, #2
22 eor r12, r12, r5, lsl #30

```

Listing 2: 优化后汇编代码实现

```

1 # 操作序列
2 r1 r0 r14 0xaa000000 0
3 r1 r0 d0 0x2a02a 2
4 r1|d0 r0 r14 0x8a00 4
5 r1|d0 r0 r14 0x2880 26
6 r1 r0 r14 0xa80000 30
7 # 汇编代码
8 uxtb.w r12, r9
9 and r5, r9, #0x2a02a
10 eor r12, r12, r5, ror #2
11 and r5, r9, #0x8a00
12 eor r12, r12, r5, ror #4
13 and r5, r9, #0x2880
14 eor r12, r12, r5, ror #26
15 and r5, r9, #0xa80000
16 eor r12, r12, r5, ror #30

```

图 3: 汇编代码实现对比分析

表 1: AES 算法实现性能对比

实现方案	优化等级	周期数	Flash 大小 (字节)
[SS16]	O3	8,932	27,100
本文工作	O3	8,068	25,948

参考文献

- [SS16] Peter Schwabe and Ko Stoffelen. All the AES you need on cortex-m3 and M4. In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 180–194. Springer, 2016.