ABY 测试文档

安装

参考 https://github.com/BintaSong/ABY

git使用

撤销commit:

- 仅撤销 commit: `git reset --soft HEAD^`
- 同时撤销 add: `git reset --hard HEAD^`
- 撤销所有更改： `git checkout -- <files>`

github加入authentication token

- `git remote set-url origin https://<username>:<token>@github.com/<repolink>`

编码

- 原码
  - 正数的原码符号位是0, 负数原码符号为是1
- 补码
  - 正数补码为就是原码，负数的补码，只需要将原码除了符号位的所有位取反后再加1。假设比特长度为k，补码本质上是在 $\mathbb{Z}_{2^k}$ 上计算。
  - ABY的算术编码于$\mathbb{Z}_{2^k}$， 其中 [0, 2^{k-1} - 1] 对应正数，[2^{k-1}, 2^k - 1] 对应负数。因此，本质是ABY的算术运算对应补码运算。

一些坑

- 对于整数 `uint64_t x`， `x << 5 + 1` 和 `x << 6` 等价，即 `+` 优先级大于 `<<` 。
- `BYTE arr[N] = {0}` 或者 `memset(arr, 0, N)` 让存储为0值。

BUGS

```
void LowMCMultiplyState(std::vector<uint32_t>& state, uint32_t lowmcstatesize, uint32_t
        std::vector<uint32_t> tmpstate(lowmcstatesize);
        for (uint32_t i = 0; i < lowmcstatesize; i++) {
                tmpstate[i] = 0;
                for (uint32_t j = 0; j < lowmcstatesize; j++) {
                        // compute current position
                        uint32_t current_pos = offset_LMatric + (round - 1) * lowmcstate
                        if (m_vRandomBits.GetBit(current_pos)) {
                                tmpstate[i] = circ->PutXORGate(tmpstate[i], state[j]);
                        }
                }
        }
        state = tmpstate;
}
```

函数 `LowMCMultiplyState` 将 `state` 代表的份额和一个 `lowmcstatesize * lomcstatesize` 的矩阵逐行做内积运算。这里，作者在循环开始时直接将 `tmpstate[i]` 初始化为0,带来错误。 `tmpstate[i] = 0` 会将0号线的值带入运算，而0号线对应值为输入的最低位（LSB）。所以，导致lowmc对于所有奇数运算出错，而对偶数输入则没有影响。 解决方法是将 `tmpstate[i]` 设置为0值输入线的ID，在 `lowmccircuit.cpp` 源码中为m_nZeroGate (即 `zero_gate = circ->PutConstantGate(0, nvals)` ，要知道，输入为SIMD，因此0值需要有 `nvals` 个)。

基本运算性能

- 在测试 benchmark和innerproduct程序时，出现安全计算和明文计算结果不一致的情况。原因是两个程序的测试实例通过rand()函数生成。

  ```
  for (i = 0; i < numbers; i++) {
      x = rand();
      y = rand();
      v_sum += x * y;
      xvals[i] = x;
      yvals[i] = y;
  }
  ```

  而rand()函数的种子为time(NULL),如果两个终端运行时获取的时间不一样，生成的测试实例不一样。
- 解决方法： 将随机数种子固定为同一个值，如 `srand(0);`

运算	协议	时间	通信量
加法	$\mathbb{Z}_{2^{32}}$ 上算术共享	4.35 s	-
乘法	$\mathbb{Z}_{2^{32}}$ 上算术共享	59.06 s	-
比较	$\{\mathbb{Z}_2\}^k$ 上比特共享	98.87 s	-

运算	协议	时间	通信量
最大值	$\{\mathbb{Z}_2\}^k$ 上比特共享	456.56 s	-
最小值	$\{\mathbb{Z}_2\}^k$ 上比特共享	496.27 s	-
方差	$\mathbb{Z}_{2^{32}}$ 上算术共享	49.069 s	-
中位数	-	-	-

1. 加法
2. 乘法
3. XOR
4. AND
5. 比较
6. 最大值
7. 最小值
8. 排序