Fundamentals of Cryptography: Project Report

Instructed by *Wenfei Wu*

Due on Jan 15, 2021

Runlong Zhou YaoClass 82 2018011309

1 Experiment Setup

1. The framework is an MPC framework called Obliv-C. I used the code contained in MPC-SoK.
2. I provided two experiments, inner product and cross tabs. Before building the framework, we should first generate corresponding test data. Enter `non-crypto` folder and choose the experiments you want, then execute

```
make gendata
./gendata
```

   Finally, copy the two input files `input*.txt` into `sok_obliv-c/source/<experiment>`.

3. The main prerequisite is `docker.io`. To build and run the framework, enter `sok_obliv-c` folder and execute

```
docker build -t obliv-c .
```

   then execute

```
docker run -it --rm obliv-c
```

4. Then, enter either `innerProd` or `crossTabs`, execute

```
make
time ./a.out 1234 -- input1.txt & ./a.out 1234 localhost input2.txt
```

   The result and executing time will be printed on console.

2 Framework Capabilities

This framework is C-style compatible.

1. Operators: addition, multiplication, bit-wise operations, etc.

2. Data type: bool, char, int, short, long, long long, float
3. Control flow: loop, if-else and any common C control flow
4. Function: supported
5. Global/local variable: the framework itself is an embedded function, so global vairables are not supported, but local variables are supported.

3 Performance

3.1 Inner product

1. **Description:** It computes inner product of two n-dimension integer vectors. Two parties each hold one vector.
2. **Result:**

n	Obliv-C	non-crypto code
10^3	$0.320\ s$	too fast
10^4	$1.962\ s$	
10^5	$17.97\ s$	$0.016\ s$
10^6	$179.3\ s$	$0.155\ s$
10^7	too slow	$1.366\ s$
10^8		$13.69\ s$

Figure 1: Execution time when n varies

From **Figure 1**, Obliv-C implements a protocol with time complexity linear in n. However, the time to setup channel takes majority when n is small, so the overall time seemes not linear in n. The performance of Obliv-C is around 100 times slower than simple, non-crypto code.

3.2 Cross tabs

1. **Description:** Party A holds a list containing (ID, category) pairs, while party B holds a list containing (ID, data) pairs. They want to compute a list of (category, sum of data in this category) pairs.
2. **Result:**

n	Obliv-C	non-crypto code
10^3	$1.013\ s$	too fast
10^4	$10.51\ s$	$0.015\ s$
10^5	$120.8\ s$	$0.175\ s$
10^6	too slow	$1.930\ s$

Figure 2: Execution time when n varies

From **Figure 2**, Obliv-C implements a protocol with time complexity almost linear in n. In fact, it implements a sort inside, so the real time complexity is $O(n \log n)$, and the non-crypto code uses three `std::map`s. This experiment shows a separation of 1000 times.