Project documentation

Lorenzo Zanolin

May 10, 2023

1 Introduction

The aim of this project is the study of Yao's protocol [3] and an useful application of it. More precisely, we will implement Secure multi-party computation; this field has the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private [2]. In this project, the function we decided to implement is the *8 bit sum*.

2 Analysis

We can focus on the sum computed by two parties, say Alice and Bob. The goal is to use *MPC* to calulate the result while avoiding each member to know the others values; to do that we need an external component, called *OT*.
This protocol allows two parties, Alice who knows x and Bob who knows y, to compute jointly the value of $f(x, y)$ in a way that does not reveal to each side more information than can be deduced from $f(x, y)$ [1]. As already written, f is the 8-bit sum.
Both Alice and Bob have a set of integers and need to compute the sum of everything. Two new roles are introduced: Alice is the *garbler* and has the responsibility of creating the garbled circuit, while Bob is the *evaluator*. They will behave as follows:

1. Alice creates the garbled circuit and the tables and send them to Bob.
2. Alice select her set, via input.
3. Bob select his set, via input.
4. Both interact with eachoter using OT.
5. Bob evaluate the function result and sends it to Alice.
6. Alice checks the correctness of the result, and sends it to Bob (in clear).

2.1 Circuit

We will present briefly the 8-bit sum circuit. There are two basic components in this construction:

- *Half Adder*: used to sum the right-most digit;
- *Full adder*: used to sum a generic digit in the number, ranging from position 1 to 8. It receives in input also carry of the previous sum.

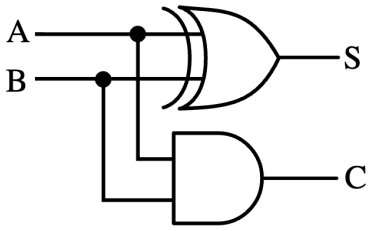

Figure 1: Half Adder

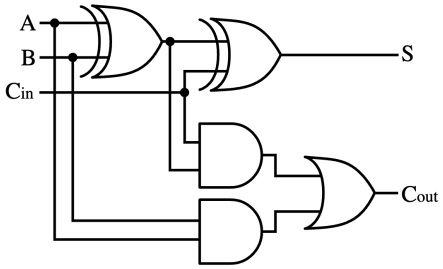

Figure 2: Full Adder

[1]

We then proceede creating the circuit by wiring 7 full adders and an half adder together, as represented in Figure 3.

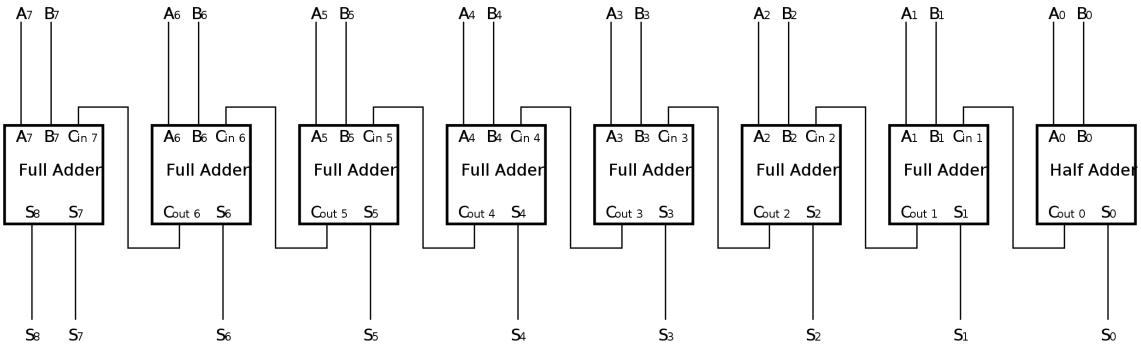

Figure 3: 8-bit Adder

[1] 1 was taken over `https://upload.wikimedia.org/wikipedia/commons/1/14/Half-adder.svg` 2 was taken over `https://upload.wikimedia.org/wikipedia/commons/a/a9/Full-adder.svg`.

2.2 Implementation

The project will be developed using *Python 3.9.10* and we will use functions provided in the GitHub repo `https://github.com/ojroques/garbled-circuit`.

2.2.1 Project structure

The project is structured as follows:

```
src/.
├── Makefile
├── images                          This directory contains the images used.
│   ├── 8-bit_full_adder.png
│   ├── Circuit.png
│   ├── Half_adder.png
│   └── Full-adder.png
├── circuits                        This directory contains the circuit used.
│   └── add.json
├── code                           This directory contains the code used.
│   ├── util.py
│   ├── yao.py
│   ├── ot.py
│   ├── requirements.py
│   └── main.py
└── sets                           This directory contains the sets saved.
    ├── alice.txt
    └── bob.txt
```

References

[1] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18, 2005.

[2] Wikipedia contributors. Secure multi-party computation — Wikipedia, the free encyclopedia, 2023. [Online; accessed 10-May-2023].

[3] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.