Introduction to Cybersecurity

Project Report

Luca Campa

1 Introduction

The project aims to provide a possible implementation of the Secure Multi-Party Computation on sets of integers throught Yao's protocol. In order to do so, I use an implementation of the Yao's protocol developed by Olivier Roques and Emmanuelle Risson[1]. In this model, Alice and Bob compute a function on their inputs without sharing their value with the opposing party. Alice is the circuit creator (the garbler) whilst Bob is the circuit evaluator. Alice creates the yao circuit and sends it to Bob along with her encrypted inputs. Bob then computes the results and sends them back to Alice [2]. The inputs from Alice and Bob will consist of sets of integers. The architecture supports different sets' cardinalities and integers of any size. The two supported functions are:

1 The sum of the values of Alice and Bob sets.

2 The common values between Alice and Bob sets.

The circuit used by Yao's protocol in order to compute the function will be dinamically build at each Alice connection to Bob. It will depend on the function we want to compute and, in particular:

- the sum will depend on the maximum length of the binary representation we are going to deal with.
- the common values (set comparison) will depend on the Alice's set cardinality and on the maximum length of the binary representation we are going to deal with.

This report is structured as follows. In Section 1 I will give you a brief introduction about the project and its purposes. Then, in Section 2, I will explain the implementation choices alongside the correctness of the implementation from a security point of view, focusing in particular on the data privacy. I will discuss about the possible use cases of the solution (real-world examples from papers I found) and I will focus on one of them from a Legal point-of-view in Section 3. Finally, I will suggest a possible improvement to the solution in order to increase its security level.
At the end I reached a solution which permits to compute one of the common problems in multi party computation: the private set intersection.

[1]https://github.com/ojroques/garbled-circuit
[2]https://github.com/ojroques/garbled-circuit/blob/master/README.md

2 The implementation

As said in the previous Section, the original implementation was taken from a GitHub repository which gives me the possibility to build my solution (based on sets of integers) on a solution based on simpler inputs.

2.1 Implementation Choices

I chose to implement two functions:

- The sum of the values from Alice and Bob sets
- The common values between Alice and Bob sets.

Bit Length In order to deal with variable integers' length, two solutions can be found:

- using a fixed maximum size (e.g. 32 bit)
- Alice will communicate with the other party in order to understand what is the maximum number of bits required to compute the result. In this way there will be less communication cost compared to the previous solution. (**MY CHOICE**)

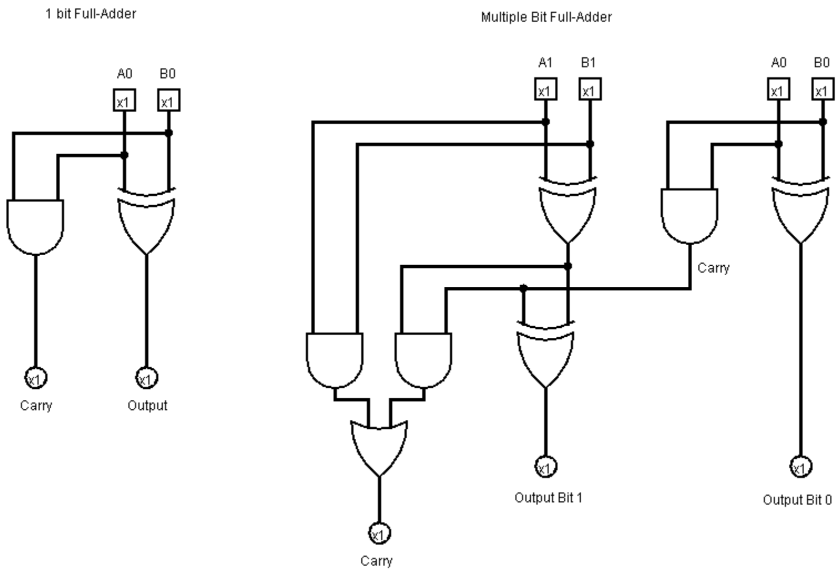

Figure 1: Sum Circuit

Set Sum Both Alice and Bob choose a set of integers. By the fact that they want to compute the sum of all these values within the sets, they precompute the sum of their respective set's values. By doing so, they will not reveal to the other party any additional information such as the number of values within their sets and the value of these integers. The circuit for the sum is built by combining multiple full-adders as you can see from the Figure 1.

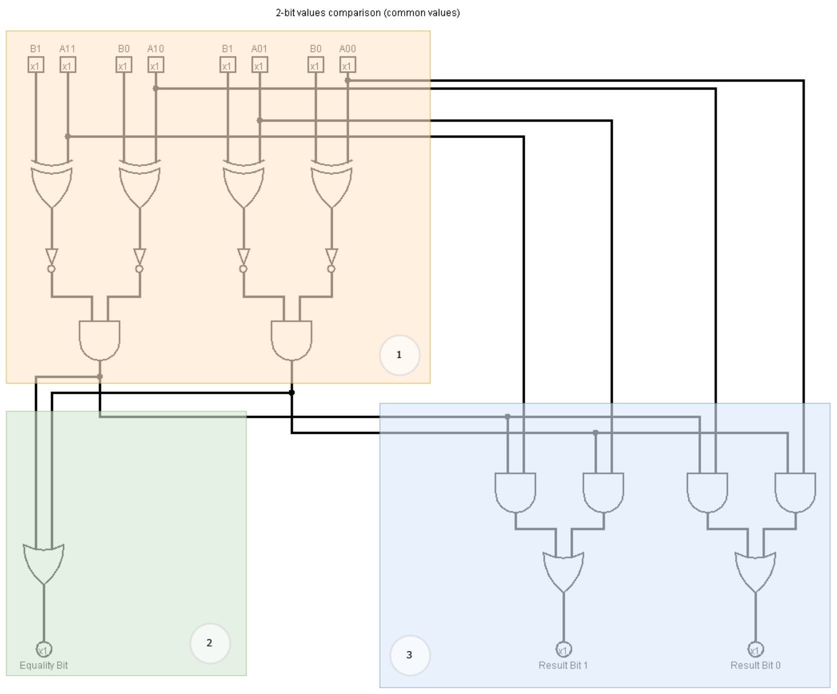

Figure 2: Common Value Circuit

Common Values (set compare) Both Alice and Bob choose a set of integers. Alice will build the circuit on the base of her set's cardinality. The circuit will receive all Alice's values and one value at each time from Bob. The value from Bob will be iteratively compared with all the Alice's inputs. The circuit will return a bit meaning the equality with one of the Alice's values alongside the common value itself. The circuit for the comparison of the sets in order to get the common value is composed by three sections combined together (Figure 2):

1 The loop for the comparison of Bob's input with each Alice's value. It returns an equality bit for each comparison.

2 The resulting equality bit from the previous ones.

3 By using the equality bits from the point (1) it returns the common value itself if exists (otherwise the bits are set to 0).

Bob will permutate his set values before sending them for the comparison.

Yao's encryption As for the encryption system used by these Yao's implementation, I chose AES mode CBC and the IV will be prepended to the ciphertext from the encryption function. An other approach could be to use the same IV for each computation but it would not be secure enough.

2.2 Security Analysis

Set Sum From a technical point of view we can analyse two main properties:

- **privacy**: it means that both the actors will not reveal information about their inputs. In my solution both compute the sum of their values:
  Given:
  - a set of m integers for Alice: $a_0, a_1, \ldots, a_m$
  - a set of n integers for Bob: $b_0, b_1, \ldots, b_n$

  $S_{bob} = sum(bob\_set) = b_0 + b_1 + \cdots + b_n$
  $S_{alice} = sum(alice\_set) = a_0 + a_1 + \cdots + a_m$

  Now they use these inputs to compute the total sum:
  $S = S_{bob} + S_{alice}$
- **correctness**: we can interpret this word in two ways:
  - the computed result through Yao's protocol is the same if computed in normal way and it is guaranteed by the method *compare_outputs* of the *ExpectedOutput* class.
  - the result received by Bob is the same received by Alice. In my solution the parties are supposed to be always honest.

Common Values As done for the previous function, we can analyse two main properties:

- **privacy**: a part from the common value, which is returned by the function, no information about the other values is provided. The first solution I wrote was supposed to send Bob's inputs on their original order.
  Given:
  - a set of m integers for Alice: $a_0, a_1, \ldots, a_m$
  - a set of n integers for Bob: $b_0, b_1, \ldots, b_n$

Each value of bob will be compared to all the Alice's values and a result of the form (0|1) || *value* will be created for each Bob's input. The result is composed by two parts: the equality bit (0: not equal, 1: equal) and the value itself. If the equality bit is 0 all the bits of the value are set to 0. Anyway, by doing it in order, it revealed to Alice an important information about Bob inputs: the position of the common values within Bob's set. Reasoning about that I came up with the following solution: it would be better to oblige Bob to permutate his values before sending them. In this way Alice will not know the position of the common values within Bob's set.
An other information which is known by Alice is the number of values inside Bob's set. This because Alice will receive a number of outputs equal to the cardinality of Bob's set. A solution to prevent this information leakage could be to:

  - create an ad-hoc hash function with a codomain of a fixed cardinality: in this way, both Alice and Bob would have a table of values with the same size (the codomain cardinality).
  - create a new circuit in which the tables are compared and the common hash values are taken and return. Each party, now, can discover on his own what are the common values.

  The proposed solution to the number of values leakage was not taken into account during the implementation.

- **correctness**: we can interpret this word in two ways:
  - the computed result through Yao's protocol is the same if computed in normal way and it is guaranteed by the method *compare_outputs* of the *ExpectedOutput* class.
  - the result received by Bob is the same received by Alice. In my solution the parties are supposed to be always honest.

3 Use cases

There are a lot of use cases of Yao's protocol applied to sets of data. Some are theoretical cases, some are real-world examples. Multi party computation became a very strong technique for securing data along computations performed from different parties which must ensure the privacy of their treated data. Examples of use cases are:

- Health:
  - Covid contact analysis: I was thinking about Immuni (the Italian application which was developed with the aim to trace the contacts between citizens affected by Covid and not).

  - Comparing a person's DNA against a database of cancer patients' DNA, with the goal of finding if the person is in a high risk group for a certain type of cancer. Such a task clearly has important health and societal benefits. However, DNA information is highly sensitive, and should not be revealed to private organisations [1].

- privacy-preserving statistics (banks, analytics) [2]

- government collaboration (finance, legislation, statistics, education,..), e.g. **Tax Fraud Detection Pilot Project**

One of the most famous practical solutions that gives the possibility to compute and perform operations in a secure way between multiple parties is Sharemind, a distributed computing system developed in Estonia to deal with secure data-sharing.

Into this section I'm going to take into account the first use case (Health) and I will try to explain how my solution could fit to it and what are the technical and legal implications.

Covid Contact Trace In the middle of 2019 Italian government asked for the development of Immuni, a mobile application for tracing contacts between Italian Citizens with the aim to contain the spread of Covid-19. Each device is associated with an ID (an ID is associated to the person once the App is installed on the smartphone). When a person conctracts Covid-19, the ID of that person is linked as Positive. How the tracking works? The Application on each smartphone checks (on a database) if the given ID or the IDs who it went in contact with became Positive.
Then, given a list of Positive IDs and a list of IDs (the personal ID and the IDs who I went in contact with), the problem is to know if one of the IDs in my list is contained into the Positive List. It is a problem similar to the one I implemented (a bit simpler because the aim is to say if there are common values and not to say what are these common values).

DNA Comparison for Cancer patients A similar problem is to compare patients' DNA with a given list with each entry composed by a DNA and an associated label (such as the type of cancer). What is this problem? It is perfectly what I tried to implement. It is an intersection between the list of DNAs I want to discover and the database of labeled DNAs. In my solution I return the common values between the two sets. Here I need to return also the associated label (the type of cancer).
This process has to deal with an interaction between a list of personal data and a list of data stored into a database. Then, there are two parties involved: the PC and the database. The goal is to secure the communication between the two parties and prevent the parties from becoming aware of the data sent by the other.

Legal Bases for the application of my solution Both the problems require to put a lot of attention to the legal bases which permit to process such kind of data. They both process data concerning people's health. In this cases, is there a legitimate interest to analyze personal health data? The Article 1 of DPA (Fundamental Right) reads:

Article 1 DPA

"[...] Such laws may provide for the use of data that, due to their nature, deserve special protection only in order to safeguard substantial public interests and, at the same time, shall provide for adequate safeguards for the protection of the data subjects' interests in confidentiality. Even in the case of permitted restrictions, a fundamental right may only be interfered with using the least intrusive of all effective methods."

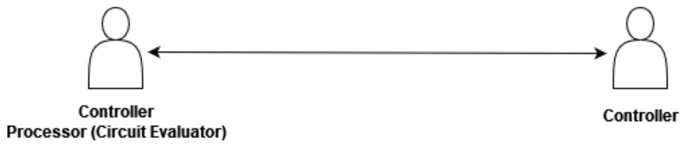

Figure 3: Actors Involved

Our goal is to preserve the health of the patient from dangerous deaseses, then, in this case, we seem to be allowed to process such kind of data. Moreover, Art. 9 of GDPR gives us the right to process such kind of data in those particular situations (see 9.2.(a), 9.2.(h) and 9.2.(i)). In particular, 9.2.(a) and 9.2(h) gives us the right to process genetic data in such situations, whilst 9.2.(a) combined with 9.2.(i) gives us the right to process Covid contacts by the fact that we are in a pandemic.

When data is taken into account for such kind of problems, we usually consider three actors: a controller, a processor and a recipient. In our model, the application of Yao's protocol MPC, we are going to have only two types of them: the controller and the processor. Indeed, we will have two controllers, one who owns patients' DNAs (or contacted people IDs) and one who owns labeled DNAs (or Covid Affected people). They will share data without revealing it by using Yao's protocol and Oblivious Transfer. Then, there will be only one processor (the circuit evaluator) which is also a controller (Figure 3).
A common technique used in order to preserve the privacy of the person whos data are going to be processed is the pseudonymisation (Art. 4(5) GDPR). The senders (the controllers) must ensure that the sent data "*can no longer be attributed to a specific data subject without the use of additional information*". [3]

[3]Article 4.(5) GDPR

Application of my solution Does my solution respect the points discussed before? Yao's protocol involves two actors, a circuit creator and a circuit evaluator. Making a link between those actors and the ones previously cited, both of them will be Controllers because they owns data to be processed. Only the circuit evaluator will be also a Processor.
My solution is meant to deal with sets of integers. The numbers can be seen as IDs, so it perfectly fits with the Covid contacts tracing. As for the DNAs' problem, DNAs can be converted into integers or numbers by using an hash function (or newer techniques - there are a lot of studies on math applied to genetic data). Moreover, the solution requires a modification because we need to return a label associated with the common value and not the common value alone.
As regards the encryption of the data the solution uses AES (mode CBC) as the cipher. GDPR requires methods to guarantee the confidentiality and the integrity of the data, then we also need to modify the solution in order to deal with it (integrity). A possible improvement could be to modify the encryption and the decryption function by implementing the *MAC-then-Encrypt* technique.

References

[1] Yehuda Lindell: **Secure Multiparty Computation (MPC)**. *IACR Cryptol. ePrint Arch.*, 2020 https://eprint.iacr.org/2020/300

[2] David W. Archer, Dan Bogdanov, Yehuda Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P. Smart, Rebecca N. Wright: **From Keys to Databases - Real-World Applications of Secure Multi-Party Computation.** *IACR Cryptol. ePrint Arch.*, 2018 https://eprint.iacr.org/2018/450