

Yao Protocol for Finding Minimum of Two Sets Using AES in Two-Party Computation

Stefano Monte

July 9, 2024

Contents

1	Introduction	2
2	Requirements	2
3	Implementation	3
3.1	Project architecture	3
3.2	Circuit	4
3.3	Functions implemented	4
3.4	Alice's pseudocode	6
3.5	Bob's pseudocode	8
3.6	Output	8
4	Usage	10

1 Introduction

This project demonstrates the secure evaluation of a function between two parties using Yao's garbled circuit protocol, aimed at implementing a MultiParty Computation (MPC) to explore its functionality and potential.

In this project, we focus on securely calculating the minimum value from sets of integers provided by Alice and Bob, ensuring that neither participant can learn the other's inputs, both during the process and in the future. The process is summarized as follows:

1. **Input Phase:** Alice and Bob each write their own set of numbers in a text file.
2. **Circuit Creation:** Alice, acting as the garbler, creates the circuit and sends it to Bob along with her encrypted inputs.
3. **Circuit Evaluation:** Bob evaluates the circuit and sends the result back to Alice.

The circuit supports at most 8-bit numbers, meaning the maximum value from each participant's set must not exceed 255 (the maximum value representable in 8 bits). This is a strict constraint: any violation will cause the model to stop and report an error.

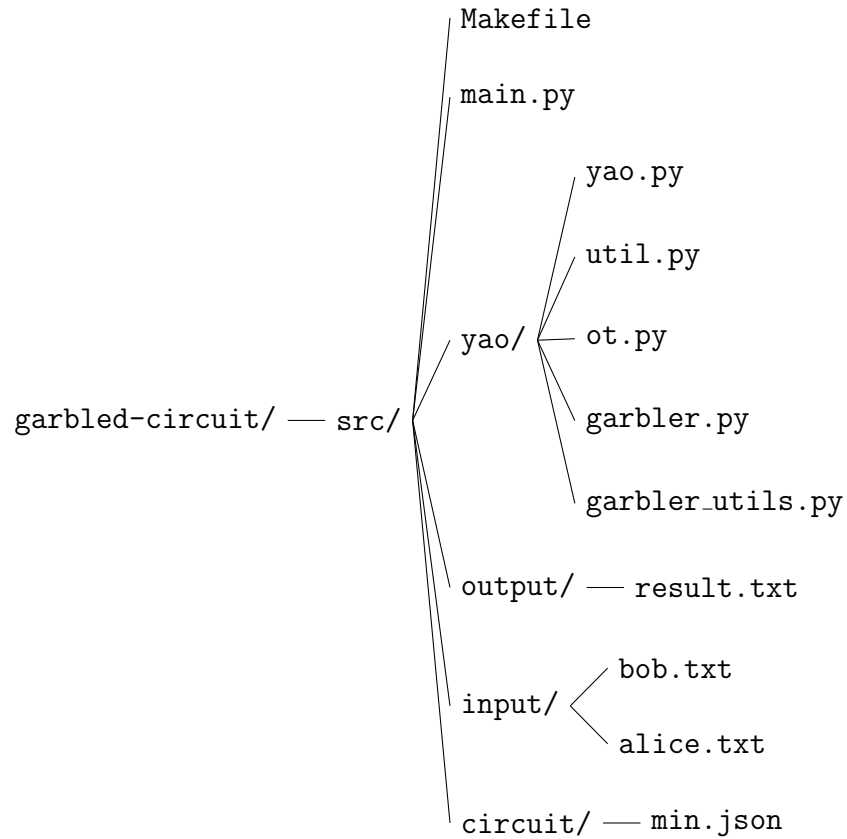
2 Requirements

The project has been developed using the python version 3.12.2 and it is based on the Olivier Roques's implementation of Yao's algorithm on Fernet's algorithm at the following GitHub repository: <https://github.com/ojroques/garbled-circuit>. The required libraries are:

- **cryptography:** provided robust and secure encryption methods, essential for safeguarding sensitive data and ensuring secure communications. With its comprehensive suite of cryptographic recipes and primitives, it facilitated the implementation of various security protocols.
- **pyzmq:** employed for messaging and communication between different components of the system.
- **sympy:** integral to the project for symbolic mathematics. This library allowed for algebraic computations, equation solving, and symbolic manipulation.

3 Implementation

3.1 Project architecture



- **circuit/:**
 - `min.json`: Contains the JSON representation of the garbled circuit.
- **input/:**
 - `alice.txt`: Contains the input values for Alice.
 - `bob.txt`: Contains the input values for Bob.
- **output/:**
 - `result.txt`: Contains the output results generated by the program.
- **yao/:**
 - `garbler_utils.py`: Contains utility functions for the core garbler script.
 - `garbler.py`: Implements the classes for Alice and Bob and manages communication between them.

- `ot.py`: Implements the oblivious transfer protocol.
- `util.py`: Provides additional utility functions.
- `yao.py`: Implements encryption and decryption functions, evaluation functions for Bob to evaluate the circuit, the `GarbledCircuit` class which generates keys, p-bits, and garbled gates of the circuit, and the `GarbledGate` class which generates the garbled table of a gate.
- `main.py`: The main script to run the program.
- `Makefile`: Contains commands for building and running the project.

3.2 Circuit

This circuit is designed to calculate the minimum between two binary numbers of size 8 bits using the principles of the Yao protocol. The circuit compares corresponding bits from the most significant bit (MSB) to the least significant bit (LSB) and determines the smallest number between the two.

Inputs:

- A7 to A0: These are the 8-bit input lines representing the first binary number.
- B7 to B0: These are the 8-bit input lines representing the second binary number.

Intermediate Logic:

- The circuit uses a combination of logic gates to process the comparisons. If the bits are equal, the comparison moves to the next significant bits. If a bit in one number is smaller than the corresponding bit in the other number, the circuit determines that number as smaller and routes the corresponding input to the output.

Outputs:

- O7 to O0: These are the 8-bit output lines that produce the final result, which is the minimum of the two input numbers.

3.3 Functions implemented

The project guidelines included a file named *format.py*, which contains several functions to be implemented. I have developed two functions that print all the details of the communications between Alice and Bob. Additionally, I created a third function to verify the correctness of the computation results.

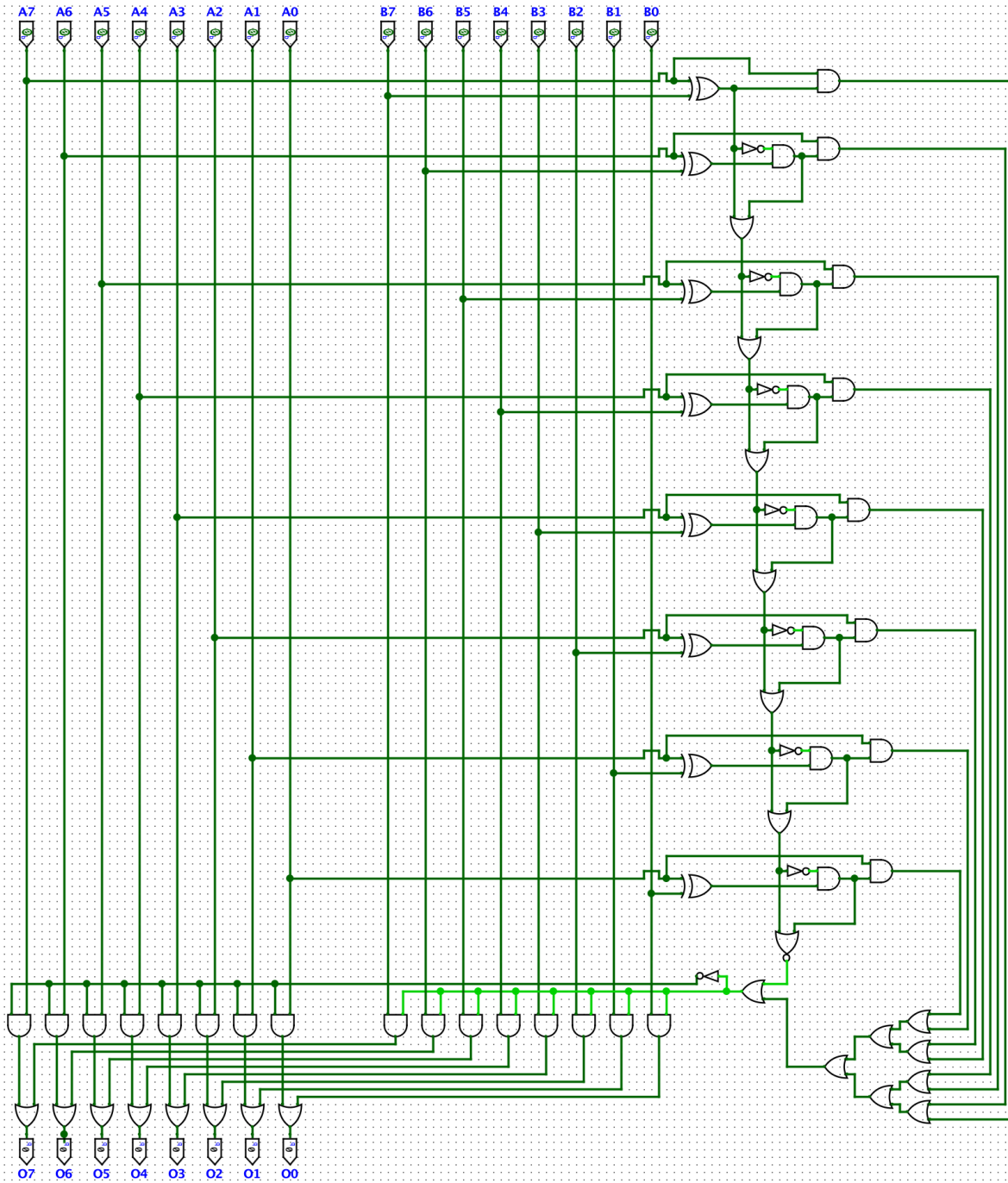


Figure 1: Circuit representation

- **alice_to_bob_OT()**: This function is designed to document and record Alice's involvement in the Oblivious Transfer (OT) process. The main objective of this function is to capture comprehensive details of Alice's computations and interactions during the OT protocol. This includes logging Alice's minimum number, the resulting message for Bob, her inputs, and the keys sent to Bob to evaluate the circuit.
- **bob_to_alice_OT()**: This function is designed to document and record Bob's involvement in the Oblivious Transfer (OT) process. The main objective of this function is to capture comprehensive details of Bob's computations and interactions during the OT protocol. This includes logging Bob's inputs, the resulting outputs, and any messages transmitted to Alice.
- **verify_output()**: This function is designed to verify the correctness of the result obtained from a garbled circuit by comparing it with a simple minimum computation performed without multiparty computation. The primary objective is to ensure the result's accuracy by validating it against independently computed minimum values from Alice and Bob's data.

There are additional utility functions for converting between decimal and binary numbers, as well as for reading from and writing to a file.

3.4 Alice's pseudocode

Data: OT, Circuit
Result: The lowest value between Alice and Bob
input \leftarrow Takes numbers from a txt file;
min \leftarrow Alice takes hers lowest value;
in \leftarrow encrypt input;
circuit \leftarrow generate and the garbled circuit;
result \leftarrow send circuit and keys to OT;
Alice writes what she sends to OT in a txt file;
Alice receives final result;

Algorithm 1: Alice side of the protocol

1. Input Reading

Alice begins by collecting her input values from a text file. These inputs are the data points she intends to use in the secure computation. After reading them, she extracts the minimum value from her set.

2. Input Encryption

Once Alice has her inputs, the next step is to encrypt these values. This process involves generating cryptographic keys for each possible value, ensuring that each input can be uniquely and securely identified.

3. Creating the Garbled Circuit

With encrypted inputs in hand, Alice proceeds to create the garbled circuit. A garbled circuit is a series of encrypted truth tables for each gate (logical operation) in the circuit. Each possible input combination for a gate is encrypted, producing an encrypted output. This step is fundamental in the Yao protocol as it allows the function to be evaluated securely.

The creation of the garbled circuit involves:

- Generating unique keys for each wire in the circuit.
- Encrypting the possible outputs of each gate using these keys.
- Constructing garbled tables that can later be used to decrypt the correct output based on the encrypted inputs.

4. Oblivious Transfer (OT)

Once the garbled circuit is ready, Alice needs to securely send it along with her encrypted inputs to Bob. This is done through the Oblivious Transfer protocol. OT is a cryptographic protocol that ensures Bob can obtain one of Alice's inputs without Alice knowing which one he chose, and without revealing the other inputs to Bob. This ensures that Bob only learns what is necessary for the computation and nothing more.

During this step, Alice sends the garbled circuit and the encrypted inputs to Bob. The protocol ensures that Bob can correctly evaluate the circuit with his inputs, obtaining the necessary intermediate results without compromising the security of Alice's inputs.

5. Receiving the Final Result

After Bob evaluates the garbled circuit using his inputs and the encrypted inputs from Alice, he obtains the result of the computation. He then sends this result back to Alice. This final result represents the outcome of the secure computation, where both parties' inputs have been used without revealing them to each other. Alice receives this final result, completing the secure computation process.

3.5 Bob's pseudocode

Data: OT

Result: The lowest value between Alice and Bob

input \leftarrow Takes numbers from a txt file (bob.txt);

min \leftarrow Bob takes it's lowest value;

data \leftarrow receive from OT;

result \leftarrow Bob calculates the final result;

Bob write his computations in a txt file;

Bob sends the results to Alice;

Algorithm 2: Bob's side of the protocol

1. Input Handling

Bob starts by reading a set of numbers from a text file, which contains his values. After reading them, he extracts the minimum value from his set.

2. Receiving Data from OT

Using the Oblivious Transfer protocol, Bob receives data from Alice. The OT protocol ensures that Bob can retrieve specific pieces of data based on his inputs without Alice knowing which pieces he selected. This is achieved through a secure cryptographic exchange, maintaining the privacy and security of the transaction.

3. Performing Computations

After receiving the data from Alice, Bob proceeds to perform the computations required to evaluate the circuit.

4. Writing the Results

After completing the computations, Bob writes the results to an output file named *output.txt*. This file serves as a record of the computed results, ensuring that they are stored securely and can be referenced later if needed.

5. Sending Results to Alice

Finally, Bob sends the results back to Alice. This step completes the communication loop initiated by the OT protocol, allowing Alice to receive the processed data or results she requires.

3.6 Output

Once both parties have input the values of their sets, the program proceeds with the application of Yao's algorithm to compute the minimum value between the two sets. An example of a correct computation is illustrated in Figure 2. At the conclusion of the computation, both parties will receive a summary containing the following information:


```

1 Bob's computation:
2
3 Bob's min number: 100 (1100100 in binary)
4 Bob's message for Alice: 00011000
5
6 Oblivious Transfer Details:
7
8 Bob's Inputs:
9 Wire 9: Bit = 0
10 Wire 10: Bit = 1
11 Wire 11: Bit = 1
12 Wire 12: Bit = 0
13 Wire 13: Bit = 0
14 Wire 14: Bit = 1
15 Wire 15: Bit = 0
16 Wire 16: Bit = 0
17
18 Bob's Output P-bits:
19 Wire 86: P-bit = 0
20 Wire 85: P-bit = 0
21 Wire 84: P-bit = 0
22 Wire 83: P-bit = 1
23 Wire 82: P-bit = 1
24 Wire 81: P-bit = 0
25 Wire 80: P-bit = 0
26 Wire 79: P-bit = 0
27
28 Alice's computation:
29
30 Alice's min number: 98 (1100010 in binary)
31 Alice's message for Bob: 01100010
32
33 Oblivious Transfer Details:
34
35 Alice's inputs:
36 Wire 1: Key = b'K2Kx6R_Lua6wf5HgP0fX7iwpCrNjNXTf8fgq0sYpbVs=', Encrypted Bit = 0
37 Wire 2: Key = b'0xV1sdXTRRQ0t0pXZvs9jHTNYB1daEhkgLEyEYeyPqbU=', Encrypted Bit = 0
38 Wire 3: Key = b'2TNWcruaIp0gadkj75DVip42W-SFyQ0TcLWH65WQ5tA=', Encrypted Bit = 1
39 Wire 4: Key = b'iyX0o56f6px13mF_hPw4U7WNWcKArULBPihDvXnJa5U=', Encrypted Bit = 1
40 Wire 5: Key = b'B7XcmdtdkWSjP_KdEv2x0FNLP0rHxtG85s-Pp651g5iU=', Encrypted Bit = 0
41 Wire 6: Key = b'SSPtMwMdwPeNMjYk-HxelztVs0IYyy73-YnG4EFG4-I=', Encrypted Bit = 1
42 Wire 7: Key = b'w2nFCp5EuXE4KiHM1WJbThRLhjcvyRYxiTwZfHtCQC0=', Encrypted Bit = 0
43 Wire 8: Key = b'GwaV6vjppPB62B1KI4T0H1iVN1Z-Tb5j6AmI06FIsg5Y=', Encrypted Bit = 0
44
45 Keys sent to Bob to evaluate the circuit:
46
47 Wire 9:
48 Key0 = muIJ8FjFh6HQoM9p0tjwIE2tY0SyxcxqQFKJmCMNqjo=, Encrypted Bit0 = 0
49 Key1 = _LUAAzs4qYaqQzjd9_gq3BI8Wz5ImLBbAZFJriMVEI=, Encrypted Bit1 = 1
50
51 Wire 10:
52 Key0 = Xi_rY0_TB6k1G1SM-W3GnZ4RM9Z594b5dLL-7JMpbg=, Encrypted Bit0 = 1
53 Key1 = BPx_goEqVM0nK2TMAzf3nLaeIfvE0otwKej_THSIgZQ=, Encrypted Bit1 = 0
54
55 Wire 11:
56 Key0 = mi8a0BgHBw58RH2BUfW2c8lei4sRABJVGlcKpd2MfiY=, Encrypted Bit0 = 0
57 Key1 = FkCK0bIyTaGghjXwc4gbE3-HOP6ASKa80wUSzgyWEJ8=, Encrypted Bit1 = 1
58
59 Wire 12:
60 Key0 = iPkgJrvemEMHMs61S6CVL2totGNq0jurqHcoT38EyaA=, Encrypted Bit0 = 1
61 Key1 = tkXhya7xvzyneQKcGHAjKHFMF05CFsgTzM1rVmQDots=, Encrypted Bit1 = 0
62
63 Wire 13:
64 Key0 = kRFZwgBq7oFIL3jCqUXNrAvicMkpIPrnVQMN-tBzqJA=, Encrypted Bit0 = 0
65 Key1 = LwM-L9XJWCMS4KkZ20pP1pbJJvRl5NE1kmlPBdgo8sU=, Encrypted Bit1 = 1
66
67 Wire 14:
68 Key0 = P0nqvQFZTLRw0k-kQEbdgz5FFsbM-Rk6f56h5vksa5o=, Encrypted Bit0 = 1
69 Key1 = bhqQENT-uDZNYxcIb5MSww10f9iBaGU0Wxc5i7C1LZ4=, Encrypted Bit1 = 0
70
71 Wire 15:
72 Key0 = GM9lecpGsz-VavzyKwI040gAWPLwYk7NEY3TBqN3IpM=, Encrypted Bit0 = 0
73 Key1 = AORHIE0bPWeLZJ5n4WSRnPB8Gk30mehVvYBq3svfV_rs=, Encrypted Bit1 = 1
74
75 Wire 16:
76 Key0 = TjRd6G8L8NU_cWqZsD5nqYLJISS2-49DtLSThittvuU=, Encrypted Bit0 = 1
77 Key1 = zc_uq4PLK91i4wM29bKZBMMye-jnL59gLbXTRTdhsYE=, Encrypted Bit1 = 0
78
79 The min of the two sets of values is 98 and it is correct.

```

Figure 2: Example of output

- **Input Values:** The initial values entered by each party.
- **Binary Representations:** The binary equivalents of the input values.
- **Encrypted Bits and Keys:** Details of the keys and encrypted bits exchanged during the oblivious transfer phase.
- **Output P-bits:** The final output bits that represent the result of the garbled circuit computation.
- **Result:** The minimum value determined from the two sets.

4 Usage

To run the code, follow these steps:

1. Prepare Input Files:

- Write Alice's input in the file named *alice.txt*.
- Write Bob's input in the file named *bob.txt*.
- Save both files in the input folder.

2. Open Terminals:

- Open two instances of the terminal.

3. Navigate to Source Folder:

- In each terminal, navigate to the *./garbler-circuit/src* directory.

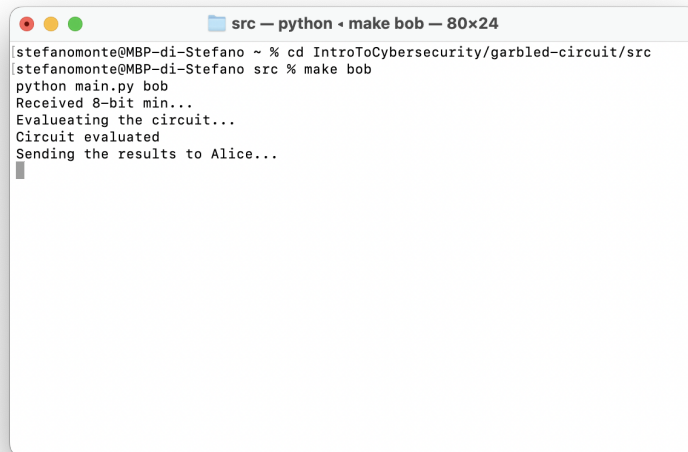
4. Install dependencies:

- Execute the command: *pip3 install -user pyzmq cryptography sympy*

5. Compile and Run:

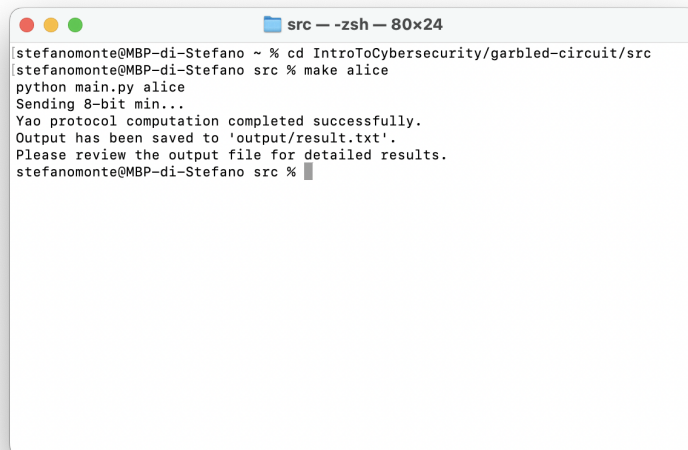
- In the first terminal, execute the command: *make bob*.
- In the second terminal, execute the command: *make alice*.

Additionally, you can print the garbler gates to the terminal by executing the *make table* command.



```
src — python · make bob — 80x24
stefanomonte@MBP-di-Stefano ~ % cd IntroToCybersecurity/garbled-circuit/src
stefanomonte@MBP-di-Stefano src % make bob
python main.py bob
Received 8-bit min...
Evaluating the circuit...
Circuit evaluated
Sending the results to Alice...
```

(a) Bob's side



```
src — -zsh — 80x24
stefanomonte@MBP-di-Stefano ~ % cd IntroToCybersecurity/garbled-circuit/src
stefanomonte@MBP-di-Stefano src % make alice
python main.py alice
Sending 8-bit min...
Yao protocol computation completed successfully.
Output has been saved to 'output/result.txt'.
Please review the output file for detailed results.
stefanomonte@MBP-di-Stefano src %
```

(b) Alice's side

Figure 3: Example of correct usage