Yao's protocol that calculates the maximum of two sets of values

Alessandro Castelli - 12246581

31-05-2023

Summary

1 Introduction

I chose to implement the maximum of two sets of values. The goal of this project is to calculate the maximum between two set of values[1] of two protagonists (Alice and Bob) considering they both don't know the other's input.

1.1 Python version and libraries

I wrote my project in Python (version 3.10.6). It is a reworking of the project that you can find at the following link: [ojroques/garbled-circuit: A two-party secure function evaluation using Yao's garbled circuit protocol (github.com)] [1].

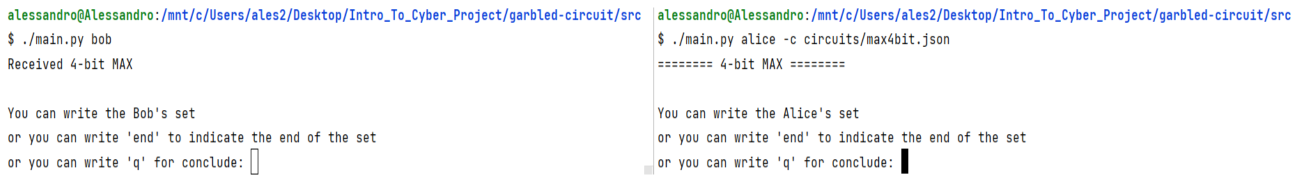
In addition to the libraries required by the above-mentioned code, it is necessary to install PyCryptodome library [2].

1.2 How to run the code

If you want to run the code, you have to follow these instructions[2]:

1. You have to open two different terminals (one for Bob and one for Alice)
2. In the Bob's terminal, you have to write "./main.py bob"
3. In the Alice's terminal, you have to write "./main.py alice -c circuits/max4B.json"

You are in the following situation (Figure 1):



```
alessandro@Alessandro:/mnt/c/Users/ales2/Desktop/Intro_To_Cyber_Project/garbled-circuit/src
$ ./main.py bob
Received 4-bit MAX


You can write the Bob's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude:
```

```
alessandro@Alessandro:/mnt/c/Users/ales2/Desktop/Intro_To_Cyber_Project/garbled-circuit/src
$ ./main.py alice -c circuits/max4bit.json
======== 4-bit MAX ========

You can write the Alice's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude:
```

Figure 1

Now, you can insert the sets of numbers (one by one).

The number has to be in the format '[d, d, d, d]' where 'd' are digits.

The digits have to be 0 or 1 because you can insert only binary numbers.

If you insert a number written in a wrong format, you can rewrite the number in the right format.

If you want to conclude the procedure for collecting inputs and if you want to see the result, you can type 'end' in both terminals.

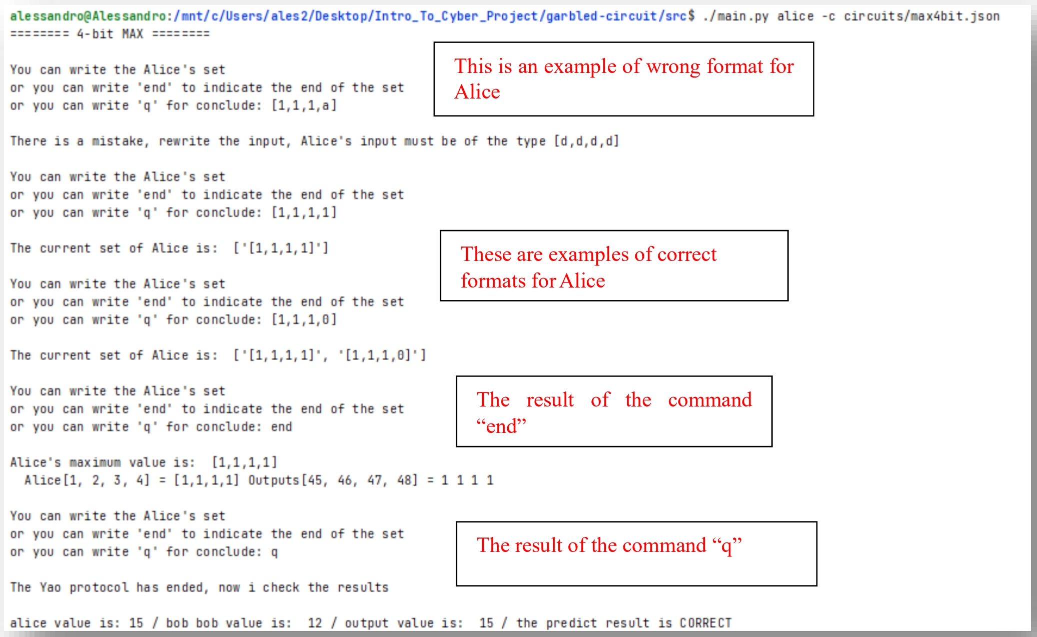
As a result, you will initially see the local maximum for Alice (in the Alice's terminal) and for Bob (in the Bob's terminal). Then you will see the result of the procedure.

Now you can start another evaluation. If you want to conclude the procedure and check the results that you have obtained until now, you can type "q" in both terminals.

---

[1] My implementation can process integers of size 4-bits
[2] I use a Linux Bash Shell

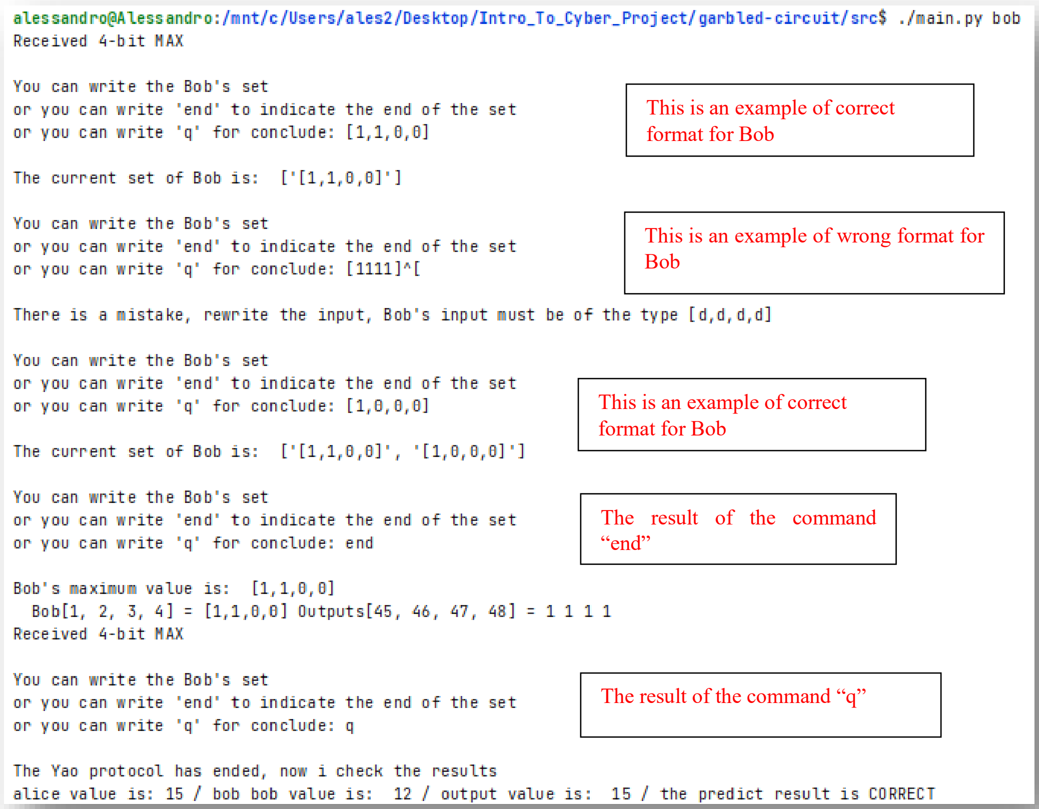
You can see an example in “Figure 2” and “Figure 3”.



```
alessandro@Alessandro:/mnt/c/Users/ales2/Desktop/Intro_To_Cyber_Project/garbled-circuit/src$ ./main.py alice -c circuits/max4bit.json
======== 4-bit MAX ========

You can write the Alice's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: [1,1,1,a]

There is a mistake, rewrite the input, Alice's input must be of the type [d,d,d,d]

You can write the Alice's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: [1,1,1,1]

The current set of Alice is:  ['[1,1,1,1]']

You can write the Alice's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: [1,1,1,0]

The current set of Alice is:  ['[1,1,1,1]', '[1,1,1,0]']

You can write the Alice's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: end

Alice's maximum value is:  [1,1,1,1]
  Alice[1, 2, 3, 4] = [1,1,1,1] Outputs[45, 46, 47, 48] = 1 1 1 1

You can write the Alice's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: q

The Yao protocol has ended, now i check the results

alice value is: 15 / bob bob value is:  12 / output value is:  15 / the predict result is CORRECT
```

This is an example of wrong format for Alice
These are examples of correct formats for Alice
The result of the command “end”
The result of the command “q”

Figure 2



```
alessandro@Alessandro:/mnt/c/Users/ales2/Desktop/Intro_To_Cyber_Project/garbled-circuit/src$ ./main.py bob
Received 4-bit MAX

You can write the Bob's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: [1,1,0,0]

The current set of Bob is:  ['[1,1,0,0]']

You can write the Bob's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: [1111]^[

There is a mistake, rewrite the input, Bob's input must be of the type [d,d,d,d]

You can write the Bob's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: [1,0,0,0]

The current set of Bob is:  ['[1,1,0,0]', '[1,0,0,0]']

You can write the Bob's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: end

Bob's maximum value is:  [1,1,0,0]
  Bob[1, 2, 3, 4] = [1,1,0,0] Outputs[45, 46, 47, 48] = 1 1 1 1
Received 4-bit MAX

You can write the Bob's set
or you can write 'end' to indicate the end of the set
or you can write 'q' for conclude: q

The Yao protocol has ended, now i check the results
alice value is: 15 / bob bob value is:  12 / output value is:  15 / the predict result is CORRECT
```

This is an example of correct format for Bob
This is an example of wrong format for Bob
This is an example of correct format for Bob
The result of the command “end”
The result of the command “q”

Figure 3

2 Circuit

I translated the following circuit[3] (Figure 4) into a json file. The name of the json file is “Max4B.json”.

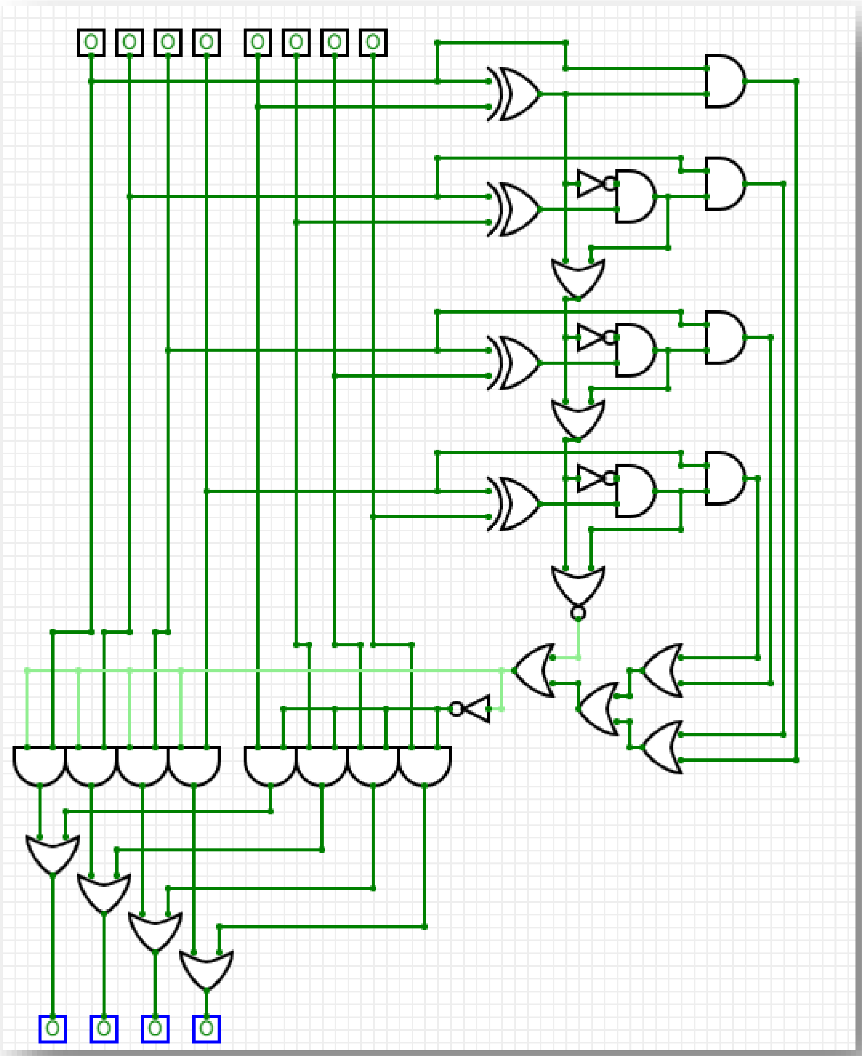


Figure 4

This circuit calculates the maximum between two binary numbers (of size 4 bits). In order to do this, the circuit compares the first two most significant bits of two numbers. If the two bits are the same, the circuit compares the second two most significant bits of two numbers and it goes on in this way until the last significant bit.

If all the bits of two numbers are the same at the end of the circuit, then the output is the same to the input. On the contrary, if one of the two numbers is greater than the other, then the output will be the bigger between the two numbers.

[3] I used CircuitVerse to draw the circuit. CircuitVerse - Digital Circuit Simulator online

3 Project development

I started by downloading the GitHub page that I found in the course page [1]. Then I changed the code according to the needs of my project.

In the following part I will explain how I changed the code and the new function that I introduced. I also wrote the pseudocode of Alice and Bob with the aim to clarify what they do.

3.1 The pseudocode of Alice's communication to Bob

```
Alice_Comunication(ObliviousTransfer, MaxCircuit){

     input ← insert Alice's input;
     write Alice's input on file;   #This is necessary for finally verification
     in ← encrypt input;
     gg ← create garbled tables;
     result_of_sending ← send_to_OT(gg, in, MaxCircuit);
     Alice receives final result;

}
```

Pseudocode 1, Alice_Comunication

3.2 The pseudocode of Bob's communication to Alice

```
Bob_Comunication(ObliviousTransfer){

     input ← insert Bob's input;
     write Bob's input on file;   #This is necessary for finally verification
     data ← receive_from_OT;
     Bob calculates the final result;
     Bob write the output of the function in an output file
     Bob sends the results to Alice;


}
```

Pseudocode 2, Bob_Comunication

3.3 Changes regarding Alice and Bob classes

I changed the class "Alice" as follows:

- Now you can insert the input from the terminal
- You can reinsert the input if you realize it is wrong
- The program calculates the local maximum of the numbers entered
- The input data is saved in a file called "alice_input.txt"
- You can write “end” to indicate the end of the set, in order to see the result (but if and only if, the other part also types "end")
- Now you have two choices:
    1. You can insert another set of input and you can restart the procedure
    2. You can write “q” to abort the procedure and then you can see the check of the result

I changed the class “Bob” as follows:

- Now you can insert the input from terminal
- You can reinsert the input if you realize it is wrong
- The program calculates the local maximum of the numbers entered
- The input data is saved in a file called "bob_input.txt"
- You can write “end” to indicate the end of the set, in order to see the result (but if and only if, the other part also typed "end")
- The output is saved in a file called “output_function.txt”
- Now you have two choices:
    3. You can insert another set of input and you can restart the procedure
    4. You can write “q” to abort the procedure and then you can see the check of the result

3.4 Encryption and decryption function

In the git repository the encryption and decryption functions were implemented using Fernet's algorithm.

The instructions required AES as encryption scheme and, for this reason, I changed them as follows.

```
def encrypt(key, data):

    IV = os.urandom(16)
    data = pad(bytes(data), BLOCK_SIZE)
    f = AES.new(key=base64.urlsafe_b64decode(key), mode=AES.MODE_CBC, IV=IV)
    return f.encrypt(data)+IV
```

Code 1, encryption function

```
def decrypt(key, data):

    IV = data[len(data)-16:]
    f = AES.new(key=base64.urlsafe_b64decode(key), mode=AES.MODE_CBC, IV=IV)
    return unpad(f.decrypt(data[:len(data)-16]), BLOCK_SIZE)
```

Code 2, decryption function

As you can see, "Code 1, encryption function" and "Code 2, decryption function" are simply an encryption function and decryption function.

I used AES with CBC mode. At this aim, I took as an example the code created during a class of "Introduction to cybersecurity".

The encryption function creates a random IV, it does the padding of the data and it finally creates the function f. The function returns the encrypted data plus the IV.

The decryption function initially extracts the IV and finally it can extract data because now it has the key and the IV.

3.5 Other functions

3.5.1 check_input

I used it to verify if the input of Alice and Bob has been inserted in the correct way. The correct format is: [d,d,d,d] where d are 0 or 1. Any other possible combination is considered wrong.

```
def check_input(input:str):

    # This function is used for check the input of Alice and Bob.
    # The correct format of the input is [d,d,d,d] where d are digits (0 or 1)
    #
    # The input is a string, the output is True if the input's format is correct, False otherwise

    if (len(input) != 9):
        return False

    elif (((input[1] != "0") and (input[1] != "1")) or ((input[3] != "0") and (input[3] != "1")) or
        ((input[5] != "0") and (input[5] != "1")) or ((input[7] != "0") and (input[7] != "1"))):
        return False

    elif (input[0] != "[") or (input[8] != "]"):
        return False

    elif (input[2] != ",") or (input[4] != ",") or (input[6] != ","):
        return False

    else:
        return True
```

Code 3, check_input

3.5.2 clean_string

I used it to change the string format. It is useful to change the format of the string during the result verification phase. This function removes brackets and commas from the input string.

```
def clean_string(a:str):
    # This function is used for clean the string
    # Examples:   [1,1,1,1] ---> 1111
    #             [1,0,1,0] ---> 1010
    #             1 1 1 1 ---> 1111
    #             1 0 1 0 ---> 1010
    if (a == ""):
        return a
    elif a[0] == "[":
        return a.replace(',', '').replace('[', '').replace(']', '')
    else:
        return a.replace(' ', '')
```

Code 4, clean_string

3.5.3 verify_output

I used it at the end of computation to verify if the procedure used in the previous steps is correct. During the computation Alice and Bob write the inputs value in a file and Bob writes the final result in an output file. This function verifies, by using the file, if it is correct.

```python
def verfiy_output(a_in, b_in, out):
    # This function is used for check the final results.

    input_alice = open(a_in, "r")
    input_bob = open(b_in, "r")
    out_max = open(out, "r")

    while True:

        a = clean_string(input_alice.readline())
        b = clean_string(input_bob.readline())
        o = clean_string(out_max.readline())

        if a == "":
            break

        a = int(a, 2)
        b = int(b, 2)
        o = int(o, 2)

        if (a > b) and (o == a):
            real_result = "CORRECT"

        elif (a < b) and (o == b):
            real_result = "CORRECT"

        elif (a == b) and (o == b):
            real_result = "CORRECT"

        else:
            real_result = "INCORRECT"

        print("alice value is:", a, "/",
              "bob bob value is: ", b, "/", "output value is: ",
              o, "/", "the predict result is", real_result)

    # Now, I can close the files
    input_alice.close()
    input_bob.close()
    out_max.close()
```

Code 5, verify_output

Bibliography

[1] [Online]. Available: https://github.com/ojroques/garbled-circuit.

[2] [Online]. Available: https://www.pycryptodome.org/.