ECE 443/518 Fall 2022 - Project 4

Garbled Circuits

Report Due: 12/04 (Sun.), by the end of the day (Chicago time)
Late submissions will NOT be graded

I. Overview

Garbled circuit enables two parties to compute a function collaboratively without revealing their own secrets. In this project, we are going to follow Lecture 18 to implement this mechanism with the following modifications.

1. Instead of using 5-bit random strings to represent 0 and 1 for wires, we are going to use 128-bit random strings to make our circuits more secure.
2. Instead of using XOR to encrypt and decrypt truth tables for gates, we are going to use the AES cipher with 256-bit key and 128-bit block. Since all of our gates are 2-input gates, this is a perfect fit - the two input signals have a total of 256 bits for the key, and the output signal, before and after encryption, has 128 bits.
3. For simplicity, we setup input signals directly instead of implementing oblivious transfer.

II. Alice the Garbler

For your conveniece, we have implement part of the garbled circuit mechanism including the garbler, input/output, and a few tests, which can be downloaded here.

The implementation of the garbler is in the file alice.go and below are the details.

- You are not supposed to modify this file (except for the bonus part).
- We number the wires in the circuit from 0 to n+m-1 where n is the number of the input bits and m is the number of the gates.
- Each Wire object contains two random 128-bit values, one for 0 and one for 1. Alice will generate these values in the function encryptWires and cannot reveal the wires array to Bob.
- Each Gate object contains the logic function, which can only be NAND for now, as well as the id's of the two input wires and the output wire. Since these values define the well-known circuit for both Alice and Bob, they are not supposed to be changed once initialized in gc.go .
- On the other hand, Alice will generate the garbled truth table for each Gate object in the function encryptGates. Since Bob will use these tables to evaluate the garbled circuit, please study this function to understand how to calculate the selection bits and how to encrypt (so decrypt) the truth table rows. Note that we use the AES block cipher directly without any mode of operations like CBC or GCM - this is different than Project 1 where the AES-GCM mode is used for AEAD.
- For 128-bit signals, we only print the first two bytes in the log messages for troubleshooting.

III. Implement Bob the Evaluator

Now it is your turn to complete the mechanism by implementing the evaluateGarbledCircuit function in bob.go . This is the only function you will need to modify for this project (excluding the bonus part). In this function, a few lines of code are provided and below are the details.

- We first create the signals array to hold one signal per wire.
- Then, the input signals are copied to the array.
- Now you need to write code evaluate the circuit by computing the gate output one by one. We setup the gates array in a way such that it is guaranteed that when you process a gate following its order in the gates array, both of its input signals are available already. As the hint in the code suggest, you will need to use a loop, and for each iteration you will need to first read the two input signals, then do some calculation, and finally update the signals array with the gate output.
- The last signal is the final output of the circuit.

IV. Testing Your Implementation

Once you have implemented the function evaluateGarbledCircuit, you are ready to test your code by running the go package using "go run ." . There are 10 testcases in gc.go which are executed from the main function. The program will abort for any error during the execution but otherwise you will be able to see the messages "test 1 pass!" etc. indicating the tests pass. Note that you are not supposed to modify this file (except for the bonus part).

V. Deliverables

Submit the following to Blackboard for this project.

1. 50 Points: Your bob.go that can pass all the 10 testcases. You will be awarded 5 pionts for each passing testcase.
2. 50 Points: A project report explaining your approach as well as screenshots of passing testcases.

VI. Bonus Tasks

Our implementation can be extended to handle more complicated cases and below are a few tasks you can work on to earn bonus points.

- Additional gate logics (20 points): Currently only NAND gates are supported. Although it is possible to implement any boolean logic with NAND gates only, it is not convenient at least and it is actually not efficient for garbled circuits. Please modify the code to support at least two kinds of logic gates from NOR, AND, OR, and XOR.
- 2-bit Comparator (20 points): Modify the code to show how Alice and Bob can compare two 2-bit unsigned integers using garbled circuit.
- 4-bit Comparator (20 points): Modify the code to show how Alice and Bob can compare two 4-bit unsigned integers using garbled circuit.
- Oblivious Transfer (40 points): Implement oblivious transfer to setup input signals for Bob. You may need to refer to Project 3 on how to use the rsa package in Go.

Please make a copy of all the source files if you decide to work on the bonus tasks as you are free to modify any files. You don't have to follow the order to complete the above four tasks. Additionally, you will need to write code to test your implementations for the bonus tasks.

Submit the code in a separate zip file to Blackboard for the bonus tasks. For each bonus task you have completed, include the following in the project report.

1. Describe your approach on how to solve it.
2. Describe your approach on how to test it and screenshots of the test results.

The project should be done individually. You can discuss the project with other students but all the source code and writings should be your OWN. PLAGIARISM and called for DISCIPLINARY ACTION. NEVER share your source code and reports with others.