CS 5990: Secure Distributed Computation Final Project

Yao's Garbled Circuits

Jared Krogsrud

This project's goal is to implement a demonstration of Yao's Garbled Circuits, in this case on an adder circuit, though it can be easily tested on any circuits that are compromised of XOR and AND gates. Yao's Garbled Circuits is a protocol in which two parties can jointly compute a circuit in which each party only has some of the inputs, in such a fashion that neither party has their inputs revealed. This implementation uses a Python notebook for easy readability and uses the PyNacl library for any encryption that need occur.

The main idea of Yao's Garbled Circuits is that the two parties involved take on the roles a Garbler and an Evaluator. The Garbler constructs garbled truth tables for every gate in a circuit, that is truth tables in which the inputs and outputs are encrypted and the ordering of the table is shuffled. The Garbler sends the Evaluator the garbled truth tables along with the encrypted values of all inputs. The evaluator then computes the circuit using the encrypted values of each wire. Once computed the Evaluator shares the resultant output wires and the Garbler decrypts and broadcasts the output of the circuit.

My specific implementation is done in 4 rounds for both the Garbler and the Evaluator. I will detail what happens in each round below:

Round 1:

Garbler: For each wire in the circuit, the garbler begins by creating a dictionary of keys for symmetric key encryption. Each wire will have two keys, one representing the wire value of 0 and the other for wire value of 1. Each gate of the circuit has a garbled truth table created for it. The truth table is created as normally but the outputs are then encrypted. For each row we look up the output wire number and it's resultant value and take that symmetric key from the previously created key dictionary. This value is then encrypted one at a time using the keys created for the input wire and value combinations.

Evaluator: The evaluator uses Oblivious transfer to send the starting values of it's wires to the garbler.

Round 2:

Garbler: The Garbler having received the public keys of the Evaluator's inputs encrypts the labels for use in circuit evaluation, sending these along with it's own keys to the evaluator.

Evaluator: Does nothing this round.

Round 3:

Garbler: Does nothing this round

Evaluator: Using all the input labels, the evaluator can now start computing the circuit. It uses the keys of the input wires and tries to decrypt the rows of the garbled truth table, only one row will successfully decrypt. This is used for the output of the gate. After all output wires are competed they are sent to the Garbler

Round 4:

Garbler: The garbler uses the output keys to lookup what their values are and broadcasts the result of the circuit computation.

Evaluator: Does nothing this round

My implementation was tested on an adder circuit with two random inputs and has successfully run every time. I decided not to try to implement this protocol for any other gate than XOR and AND as conventionally these are the two gates that the circuits tend to be constructed from.