Secure Two Party Computation

Preliminary presentation

Nick Tutte

Prof. Nigel Smart

20$^{\text{th}}$ February, 2015

Presentation overview

- My project focuses on Secure Multi-party Computation, in particular the two party case using Yao Garbled Circuits.
- The final aim is to implement the as yet unimplemented protocol laid out by Lindell in "Fast Cut-and-Choose based Protocol for Malicious and Covert Adversaries" (henceforth Lindell 2013).
- By the end of this presentation you should know,
  - What is Secure Multi-party Computation?
  - What can it be used for?
  - The basics of Yao Garbled Circuits.
  - What progress have I made so far.

What is Secure Multi-party Computation?

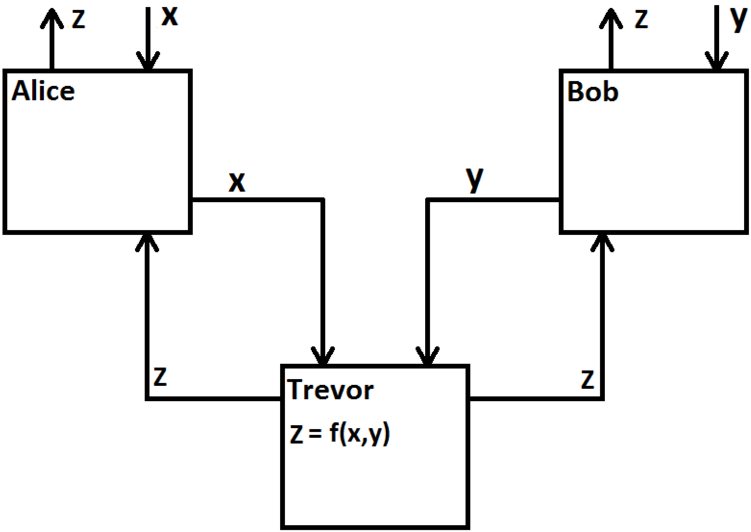
- In the problem of Secure Multi-party Computation we have a set of parties, each of whom has a secret input.
- The parties wish to co-operate to compute a function upon their inputs without revealing said inputs to one another.
- Some example applications are,
  - The Millionaires problem.
  - Sugar Beets (Auctions).
  - Distributed secrets.

Desired security properties

Before we go any further we need to define what security properties we want an SMC protocol to have.

- **Privacy**, the only knowledge parties gain from participating is their proper outputs.
- **Correctness**, the output is indeed that of the intended function.
- **Independence of inputs**, no party can choose it's inputs as the function of other parties inputs.

The Ideal Model



Security Definitions

- We measure the security of an SMC protocol in terms of what adversaries it is maintains the aforementioned properties against.
- We define adversaries in terms of their capabilities.
- We say that an SMC protocol is secure against an adversary if the adversary can achieve no more than they would be able to achieve attacking the Ideal Model.
- The three main adversaries we consider are,
    - Semi-Honest
    - Malicious
    - Covert

Semi-Honest Adversaries

- Semi-Honest(SH) adversaries are the weakest adversary we shall consider.
- They are sometimes also called "honest, but curious".
- SH adversaries are limited to looking at information given to them in the process of the protocol.
- They have to follow the protocol (they cannot cheat).
- SH adversaries are very similar to traditional "Passive" adversaries.

Malicious and Coverts Adversaries

- Malicious adversaries are the strongest adversary.
- Malicious adversaries can use any arbitrary strategy. We do not assume that they follow the protocol.

- Covert Adversaries are slightly weaker than Malicious Adversaries.
- Covert Adversaries can also use any arbitrary strategy, but they are adverse to being caught.
- They are willing to accept a certain probability of getting caught.

Oblivious Transfer

- A key component we will need later is Oblivious transfer(OT).
- Security definitions for OTs is are very similar to SMC, but we won't dwell on this too much here.

Receiver	**Sender**
Inputs : $b \in \{0, 1\}$	Inputs : $X_0, X_1 \in \{0, 1\}^t$
Outputs : X_b	Outputs : $\emptyset$

Figure 1 : Definition of the functionality of a one-out-of-two OT protocol. Note k-out-of-n OT is also possible.

Even-Goldreich-Lempel Semi-Honest OT

Receiver
Inputs : $b \in \{0, 1\}$
Outputs : X_b

Sender
Inputs : $X_0, X_1 \in \{0, 1\}^t$
Outputs : $\emptyset$

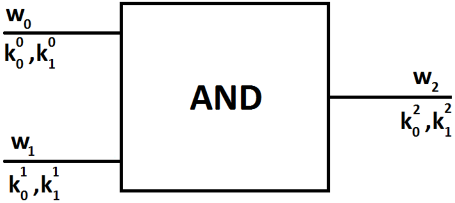
- **Receiver:** Generates a public/private key pair (E, D), where E is the public key.
- **Receiver:** Sets $PK_b = E$, choose PK_{1-b} at random from the same distribution as the public keys. Send PK_0 and PK_1 to the Sender.
- **Sender:** Encrypt X_0 using PK_0 as C_0 and encrypt X_1 using PK_1 as C_1. Send C_0 and C_1 to the Receiver.
- **Receiver:** Receives C_0 and C_1, then decrypt C_b using D. Output this decrypted value.

Figure 2 : The abstracted Even-Goldreich-Lempel protocol. Who can suggest why this is only Semi-Honest?

Yao Garbled Circuits

- In 1986 Andrew Yao proposed a protocol for S2C based on obfuscating binary circuits.
- The the binary circuit representing the function to be computed is "garbled".
- We denote the value of wire w_i as b_i, we generate two random garble values/keys, denote these by k_i^0 and k_i^1.
- We then generate a random 1-bit permutation for each wire w_i, denote this by π_i.
- For each gate we create an encryption table indexed by c_i and c_j (where the input wires to the gate are w_i and w_j).

Yao Garbled Circuits



$$c_0, c_1 : E_{k_0^{b_0}}(E_{k_1^{b_1}}(k_2^{G(b_0,b_1)} \| c_2))$$

where $c_i = \pi_i(b_i)$, $G(b_i, b_j)$ and $c_2 = \pi_2(G(b_0, b_1))$ is the function taking the input of the gate and returning the output of the gate.

Yao Garbled Circuits

- We extend this to all gates of the circuit.
- The Builder then sends the circuit to the Executor, stripped of the values of the permutations and keys.
- The Builder then sends the keys and permuted bit values relating to the its inputs (e.g. if the Builder's i^{th} input is 0, the builder sends k_i^0 and $\pi_i(0)$).
- The Builder also sends the permutations for the Executor's input wires and the Executor's output wires.
- The Executor then obtains the keys for its inputs by running Oblivious transfers with the Builder.
- Finally the Executor uses the keys evaluate the circuit.

Yao Garbled Circuits - Malicious Security

- Naively implemented Yao Garbled Circuits are only Semi-Honest secure.
- Can anyone suggest why this might be?

Yao Garbled Circuits - Malicious Security

- Yao Garbled Circuits are only secure up to Semi-Honest adversaries because we are trusting the circuit Builder to build the correct circuit.
- There are several ways to extend Yao Garbled Circuits to achieve security in the presence of Malicious adversaries.
- In particular let's look at "Cut-and-choose", a method loosely inspired by the solution to dividing a cake evenly between two parties.
- One person cuts the cake into two halves, the other person chooses which half they want.

Cut-and-Choose

- In Cut-and-choose the Builder generates many circuits.
- The Executor then picks a random subset of the circuits and "opens" them, to check that they are correct.
- If any of the "Check-Circuits" fail the Executor knows the Builder is trying to cheat.
- The Executor then evaluates all the remaining un-opened circuits and returns the majority output.

Cut-and-Choose - Not so simple

- Cut-and-choose seems like an incredibly simple solution to solve all our problems, but it creates several new problems.
- For example we need to ensure the Builder gives the same input for every circuit, without knowing what their input should be!
- The Executor also has to careful about how they react to detecting cheating, they can't just immediately abort.

Why Lindell 2013 is Great

- The best result for a Cut-and-choose based protocol before Lindell 2014 gave an error rate of $2^{-0.32s}$ where s is the number of circuits.
- Lindell's new protocol gives an improved error rate of 2^{-s}.
- This means we can get the same level of security for three times less circuits.

Questions?