

# Secure Two Party Computation

## Preliminary presentation

Nick Tutte

Prof. Nigel Smart

February, 2015

# Presentation overview

- ▶ My project focuses on Secure Multiparty 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 2014).
- ▶ By the end of this presentation you should know,
  - ▶ What is Secure Multiparty Computation?
  - ▶ What can it be used for?
  - ▶ What “Secure” means in this context.
  - ▶ A grounding in Yao Garbled Circuits.
  - ▶ How much progress I’ve made so far.

# What is Secure Multiparty Computation?

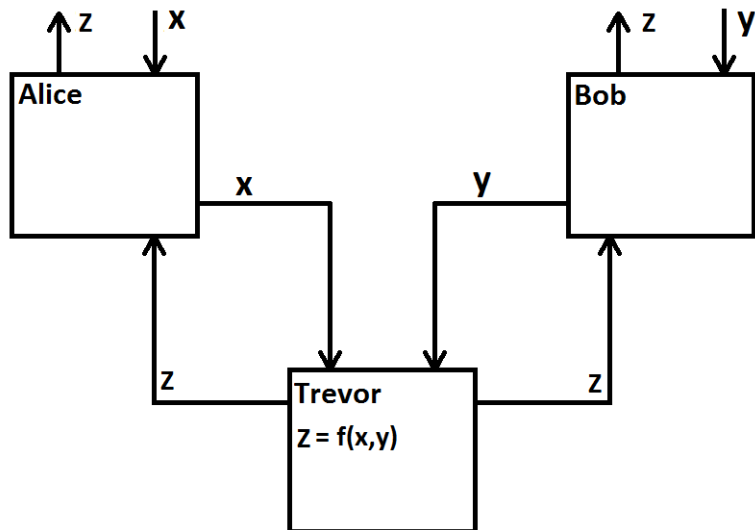
- ▶ In the problem of Secure Multiparty Computation we have a set of parties, each of whom has a secret input.
- ▶ The parties wish to co-operate to compute a function upon their collective inputs without revealing said inputs to one another.
- ▶ Some example applications are,
  - ▶ The Millionaires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

# Desired security properties

Before we go any further we need to define what properties we want an SMC protocol to fulfill before we consider it Secure.

- ▶ **Privacy**, the only knowledge parties gain from participating is the output.
- ▶ **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 secure 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.
- ▶ We focus on three adversaries,
  - ▶ 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, though we don't really look at the Covert case.

## Receiver

Inputs :  $b \in \{0, 1\}$

Outputs :  $X_b$

## Sender

Inputs :  $X_1, X_2$

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 :  $X_0, X_1$

Outputs :  $\emptyset$

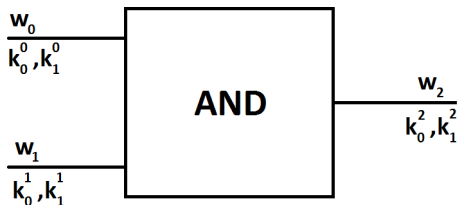
- ▶ **Receiver:** Generates a public/private key pair  $(E, D)$ , where  $E$  is the public key.
- ▶ **Receiver:** Sets  $PK_b = E$ , choose  $PK_{1-b}$  at random from the same distribution as the public keys. Send  $PK_0$  and  $PK_1$  to the Sender.
- ▶ **Sender:** Encrypt  $X_0$  using  $PK_0$  as  $C_0$  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

- ▶ The basic concept of Yao Garbled Circuits is that one party constructs a binary circuit corresponding to the function to be computed.
- ▶ For wire  $w_i$  we denote the value of the wire as  $b_i$ , we generate two random “garble value”, denote these by  $k_i^0$  and  $k_i^1$ .
- ▶ We then generate a random permutation for each wire  $w_i$ , denote this by  $\pi_i$ .
- ▶ For each gate we create an encryption table indexed by  $c_i$  and  $c_j$  (where the gates input wires 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  $c_i = \pi_i(b_i)$  and  $G(.,.)$  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 relating to its inputs for its input wires.
- ▶ The Builder also sends the permutations for the Executors input wires and the Executors 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 Garble Circuits - Malicious Security

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

# Yao Garble Circuits - Malicious Security

- ▶ Yao Garbled Circuits are only secure up to Semi-Honest adversaries because we are trusting that the Circuit builder is building the correct circuit.
- ▶ There are several ways to extend Yao Garbled Circuits to achieve security in the presence of Malicious adversaries.
- ▶ In particular I'm looking 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 give the correct result.
- ▶ 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.
- ▶ The number of circuits the builder creates is a security parameter and defines how likely a cheating adversary is to remain undetected.



# 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 be careful about how they react to detecting cheating, they can't just immediately abort.

# Why Lindell 2014 is Great

- ▶ The best result for a Cut-and-choose based protocol before Lindell 2014 was

# Questions?