

# Chaining Garbled Circuits

Alex Ledger

Reed College

April 3, 2016

# Overview

1. Secure Computation
2. Garbled Circuits
3. Better Garbled Circuits
4. Chaining Garbled Circuits
5. Better Chaining of Garbled Circuits

# Goals of this talk

1. Understand the high level idea of secure computation
  - ▶ Maybe you'll run into a situation where it's useful, and it can help solve an otherwise unsolvable, real world problem.
2. Understand garbled circuits - the most basic construction
3. Understand chaining - how we (my thesis) made secure computation faster
4. We will not be focusing on security (different from most crypto talks)

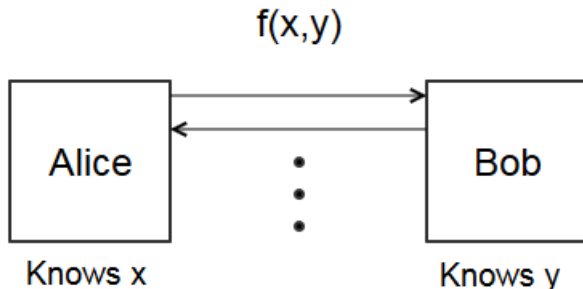
Before we start:

1. Lots of notation: ask me if I brush over something, or you forget what something means.
2. Lots of moving parts.

# The Millionaire Problem

- ▶ Alice and Bob want to determine who is wealthier.
- ▶ They do want to disclose their wealth to each other.
  - ▶ Alice has \$ $x$ , Bob has \$ $y$ .
  - ▶ Alice should not learn anything about  $y$ .
  - ▶ Bob should not learn anything about  $x$ .

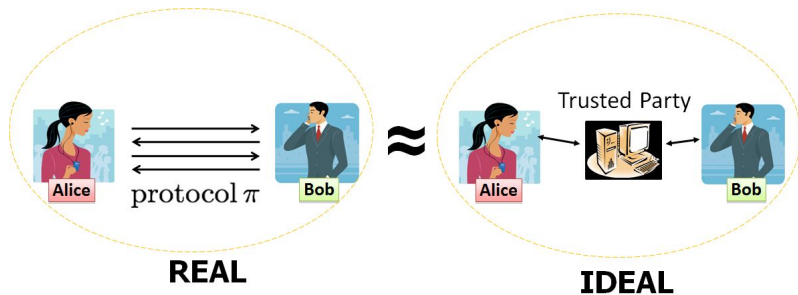
$$f(x, y) = \begin{cases} \text{Alice}, & y \leq x; \\ \text{Bob}, & y > x. \end{cases} \quad (1)$$



# Security Properties

- ▶ Privacy of inputs
  - ▶ Alice and Bob do not learn anything about the other's input.
  - ▶ Except for info that is inferable from  $x$  and  $f(x, y)$ .
  - ▶ Bob should not learn that  $1,000,000 < x \leq 2,000,000$ .
  - ▶ But if  $y < x$  and  $y = 2,000$ , then he learns  $x < 2,000$ .
- ▶ Correctness
  - ▶ Alice and Bob receive  $f(x, y)$ .
  - ▶ As opposed to some value near  $f(x, y)$ .
  - ▶ Or not receiving a value at all
- ▶ Semi-honest
  - ▶ We assume that each party obeys the protocol, but attempts to learn extra information from its interactions

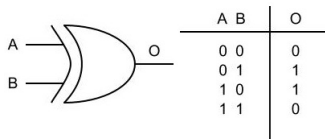
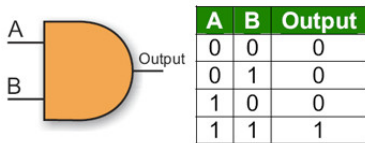
# Security



A secure computation protocol is secure if Alice and Bob learn the same information in the real world as the ideal world.

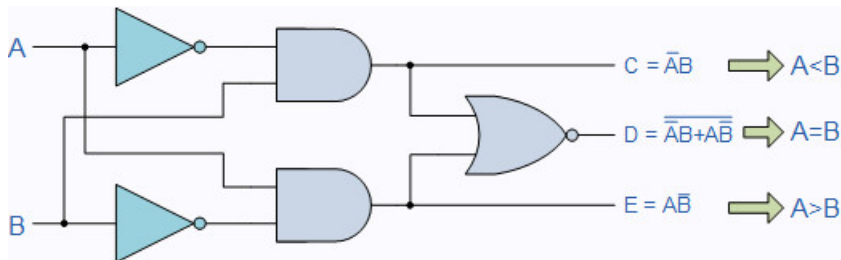
# Boolean Circuits

- ▶ We encode a function  $f$  into a circuit  $C$ .
- ▶ Circuit  $C$  is made of AND, XOR and NOT gates.
- ▶ Each gate has two input wires and a single output wire



# Boolean Circuits

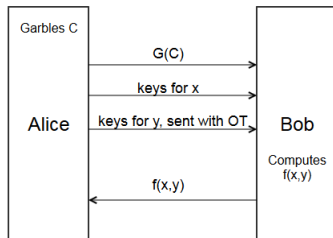
- ▶ Any function can be encoded into a circuit.
- ▶ Here is the less than circuit.





# Roadmap of Garbled Circuits

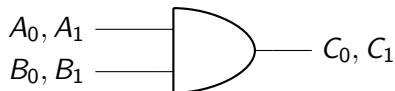
1. Suppose Alice and Bob are computing a circuit that is only the AND gate.
2. Alice *garbles* the AND gate.
3. Alice then sends *garbled table* of the gate and some auxiliary information to Bob.
4. Bob *evaluates* the gate.



# Garbling a gate 1

Step 1. Assign *wire labels* to each wire.

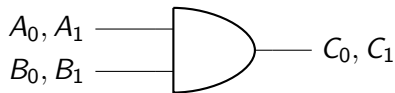
- ▶ For each wire in the circuit, assign two random labels to each wire
- ▶ Wire  $A$  has wire labels  $A_0$  and  $A_1$ .
- ▶ We say  $A_0$  *semantically represents* 0, and  $A_1$  *semantically represents* 1.
- ▶ And  $A_0$  and  $A_1$  are sampled uniform randomly from  $\{0, 1\}^n$ .
  - ▶ Generally use AES-128 for encryption, so  $n = 128$ .



## Garbling a gate 2

Step 2. Construct garbled table.

- ▶ Encrypt wire labels for  $C$ ,  $C_0$  and  $C_1$ , using wire labels of  $A$  and  $B$ .
- ▶ Randomly permute table



$A$	$B$	Encryption
$A_0$	$B_0$	$\text{Enc}_{A_0, B_0}(C_0)$
$A_1$	$B_0$	$\text{Enc}_{A_1, B_0}(C_0)$
$A_0$	$B_1$	$\text{Enc}_{A_0, B_1}(C_0)$
$A_1$	$B_1$	$\text{Enc}_{A_1, B_1}(C_1)$

## Garbling a gate 3

Step 3. Send garbled table to Bob.

$Enc_{A_0, B_0}(C_0)$
$Enc_{A_1, B_0}(C_0)$
$Enc_{A_0, B_1}(C_0)$
$Enc_{A_1, B_1}(C_1)$

## Garbling a gate 4

Step 4. Send input wire labels to Bob.

- ▶ Suppose  $x_a, x_b \in \{0, 1\}$  are Alice's inputs.
- ▶ Alice sends Bob  $A_{x_a}$  and  $B_{x_b}$ 
  - ▶ The wire labels corresponding to her inputs.

Garbled Table
$\text{Enc}_{A_0, B_0}(C_0)$
$\text{Enc}_{A_1, B_0}(C_0)$
$\text{Enc}_{A_0, B_1}(C_0)$
$\text{Enc}_{A_1, B_1}(C_1)$

Input Labels
$A_a$
$B_b$

# Garbling a gate 5

Step 5. Bob evaluates the circuit

- ▶ Bob has the garbled table,  $A_{x_a}$  and  $B_{x_b}$ .
- ▶ Bob trial decrypts each row of the garbled table, until an encryption succeeds.
- ▶ Bob acquires  $C_{a \wedge b}$ .

Garbled Table
$\text{Enc}_{A_0, B_0}(C_0)$
$\text{Enc}_{A_1, B_0}(C_0)$
$\text{Enc}_{A_0, B_1}(C_0)$
$\text{Enc}_{A_1, B_1}(C_1)$

Input Labels
$A_a$
$B_b$

Output Label
$C_{a \wedge b}$

# Garbling a gate 6

Step 6. Bob gets a final answer.

- ▶ Alice sends Bob  $\text{Enc}_{C_0}(0)$  and  $\text{Enc}_{C_1}(1)$ .
- ▶ Bob trial decrypts these with  $C_{a \wedge b}$ .
- ▶ One will succeed, and Bob will acquire  $c = a \wedge b$ .
- ▶ So Bob knows  $c$ , but not  $a$  and  $b$ !

Garbled Table
$\text{Enc}_{A_0, B_0}(C_0)$
$\text{Enc}_{A_1, B_0}(C_0)$
$\text{Enc}_{A_0, B_1}(C_0)$
$\text{Enc}_{A_1, B_1}(C_1)$

Input Labels
$A_a$
$B_b$

Output Label
$C_{a \wedge b}$

Output Map
$\text{Enc}_{C_0}(0)$
$\text{Enc}_{C_1}(1)$

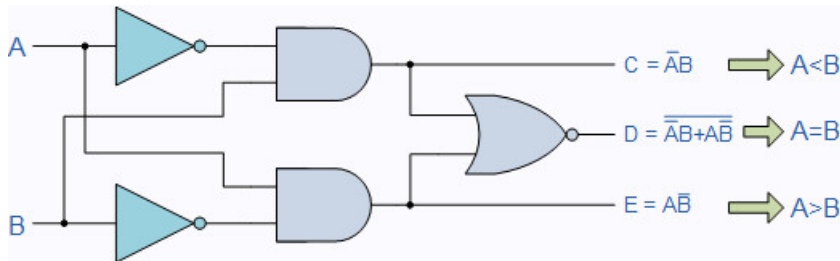
# Security Considerations

- ▶ Think about what Alice acquires:
  - ▶ Well Bob had no input in this case, so ...
  - ▶ In general, Alice generates objects and sends them to Bob
  - ▶ So she doesn't have many opportunities to learn about Bob's input.
- ▶ Think about what Bob acquires:
  1. Garbled table
  2. Input wire labels:  $A_a$  and  $B_b$
  3. Encryptions of output:  $\text{Enc}_{C_0}(0)$  and  $\text{Enc}_{C_1}(1)$
- ▶ Can Bob learn anything about  $a$  or  $b$ ?
  - ▶ If he could decrypt another row of the garbled table, then we would learn something.
    - ▶ But he can't, because he doesn't have the keys.
  - ▶ Not really, because everything is encrypted.



# Extending a garbled gate into a garbled circuit

- ▶ To operate on a more complex function, the operation is recursed.
- ▶ The output wires of the first gates are used as inputs to subsequent gates.
- ▶ Alice only sends output maps for the final gates.



# The cost of garbled circuits

- ▶ Alice sends Bob 4 ciphertexts per gate, since the garbled table has 4 rows.
- ▶ This is the bandwidth, or the amount of the communication required.
- ▶ Based on empirical work, bandwidth is the biggest bottleneck in garbled circuits.
- ▶ So reducing the size of the garbled table is of utmost priority.

# Reducing the size of the garbled table with Free XOR

- ▶ Most improvements to garbled circuits are about choosing wire labels intelligently.
- ▶ Let  $\Delta \leftarrow \{0, 1\}^n$  be fixed globally in a circuit.
- ▶ Let  $W_0 \oplus W_1 = \Delta$  for all wires.
  - ▶ aka  $W_1 = W_0 \oplus \Delta$ .
- ▶ Then an XOR gate can be computed by XORing the two input labels.

$$A_0 \oplus B_0 = C_0$$

$$A_0 \oplus B_1 = A_0 \oplus (B_0 \oplus \Delta) = (A_0 \oplus B_0) \oplus \Delta = C_1$$

$$A_1 \oplus B_0 = (A_0 \oplus \Delta) \oplus B_0 = (A_0 \oplus B_0) \oplus \Delta = C_1$$

$$A_1 \oplus B_1 = (A_0 \oplus \Delta) \oplus (B_0 \oplus \Delta) = (A_0 \oplus B_0) = C_0$$

- ▶ So XOR gates do not require a garbled table.
- ▶ They are free
- ▶ Secure?

# Introducing Online/Offline

- ▶ Imagine two banks use secure computation during their daily operations
- ▶ At night - the offline phase - they exchanged garbled circuits (the garbled tables)
- ▶ During the day - the online phase - they exchange input wire labels and use the pre-exchanged garbled circuits.
- ▶ This is better for the banks, as the computation performs faster and isn't a burden on them.
- ▶ Pitfalls
  - ▶ Functions are decided at night - no room for flexibility
  - ▶ Input size is fixed at night
    - ▶ rectifiable if the input is padded, but that's inefficient and annoying

# Chaining Garbled Circuits

- ▶ Goal: improve offline/online garbled circuits by adding flexibility
- ▶ Key observation: many useful functions in the real world are constructed in a modular way
  - ▶ They are composed of standard components
  - ▶ E.g. addition, subtraction, matrix multiplication
- ▶ Idea: chain garbled circuits together
  - ▶ Take the output of one garbled circuit and plug it into another garbled circuit
- ▶ ADD IMAGE

# How to chain garbled circuits

- ▶ Suppose that we are chaining a garbled circuit with output wire  $A$  to garbled circuit with input wire  $B$ .
- ▶ We want  $A_0 \rightarrow B_0$  and  $A_0 \oplus \Delta \rightarrow B_0 \oplus \Delta$ .
- ▶ Straightforward:
  - ▶ Alice sends Bob  $\Gamma = A_0 \oplus B_0$
  - ▶ Bob sets  $B_*$  to  $A_* \oplus \Gamma$ .
  - ▶ TODO haven't introduced  $*$  star notation.
- ▶ Is this secure? What does Bob learn?
  - ▶ Not really anything.

# Efficiency of chaining garbled circuits

- ▶ In online phase, communicate a ciphertext per chain
- ▶ This can be a lot:
  - ▶ Suppose 100 by 100 matrix with entries of max value  $2^8$
  - ▶ Then requires 256,000 ciphertexts.
  - ▶ This can be reduced to a single ciphertext.

# Single Communication Multiple Connections SCMC

- ▶ Problem: number of ciphertexts communicated scales linearly with the number of chains
- ▶ Key observation: the chaining is predictable, consecutive wires are chained to consecutive wires
- ▶ TODO SHOW EXAMPLE OF CONSECUTIVE WIRES
- ▶ Solution: give consecutive wire labels a predictable pattern while maintaining the Free XOR condition
- ▶ Set the  $i$ th wire label to  $A \oplus H(T \oplus (i||b))$ 
  - ▶ Where  $H$  is a hash function
  - ▶ TODO figure out what to do about hash function



# Single Communcation Multiple Connections SCMC

