Multiparty Computation (MPC)

Daniel Kales

Applied Cryptography 2 – ST 2020

Outline

Introduction to Multiparty Computation

Cryptographic Primitives

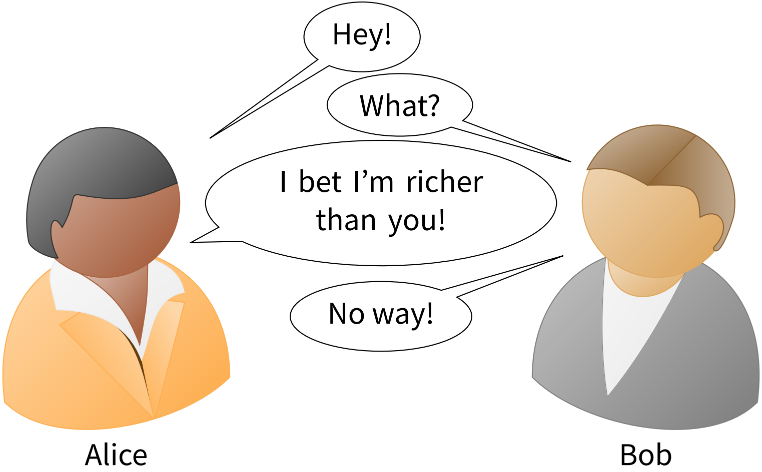
- Preliminaries
- 1-out-of-2 Oblivious Transfer
- 1-out-of-N Oblivious Transfer

Protocols for Multiparty Computation

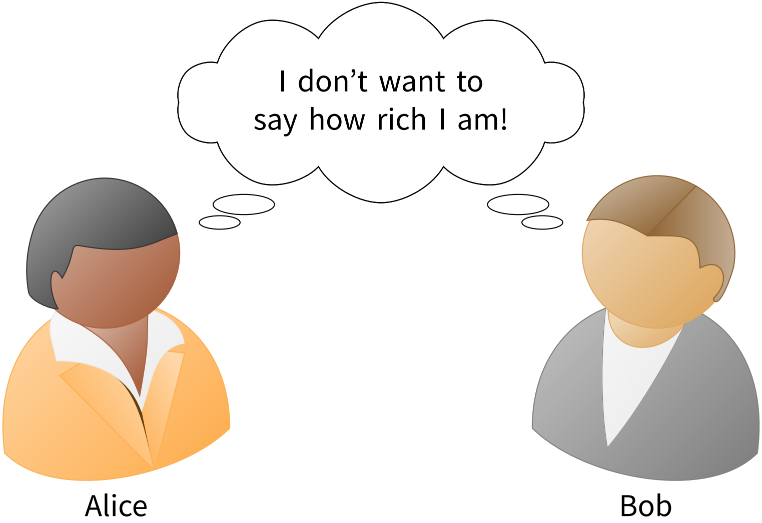
- Yao's Garbled Circuits
- Goldreich-Micali-Wigderson

Introduction to Multiparty Computation

Yao’s Millionaires’ Problem [6]



Yao's Millionaires' Problem [6]



Secure Function Evaluation

- Generalization of the Problem
  - j parties
  - Inputs: $x_i \in \{0,1\}^n$
  - Function: $F : \{0,1\}^{jn} \rightarrow \{0,1\}^m$
- Construct protocol that ensures:
  - Some/All Users learns $F(x_1, \ldots, x_j)$
  - User i learns nothing about x_j for $j \neq i$
  - User i learns nothing about any intermediate values

Secure Function Evaluation (cont.)

- Applied to the Millionaires' Problem
  - 2 parties: Alice and Bob
  - Inputs: $x_1, x_2 \in \{0, 1\}^{32}$ (32-bit integers)
  - Function: $F(x_1, x_2) = x_1 > x_2$

Secure Function Evaluation (cont.)

- Trivial Solution
  - Use trusted third party
  - Each party sends it's input and desired function to TTP
  - TTP calculates the result and sends it to parties
- Can we do it without a TTP?

Secure Function Evaluation (cont.)

- Trivial Solution
  - Use trusted third party
  - Each party sends it's input and desired function to TTP
  - TTP calculates the result and sends it to parties
- Can we do it without a TTP?
  - Yes!
  - Yao: Garbled Circuits [5]
  - Goldreich,Micali,Wigderson: GMW protocol [3]
  - More on these later…

Cryptographic Primitives

Preliminaries

Reminder: Groups

Definition (Group)

An Abelian group $\langle S, *\rangle$ is a set S and an operation $*$ that satisfy

1. Associative: $a * (b * c) = (a * b) * c$
2. Commutative: $a * b = b * a$
3. Neutral element (identity) e: $a * e = a$
4. Inverse element a^{-1} for every a: $a * a^{-1} = e$

A finite group is a group with a finite number of elements.

Examples:

$\langle \mathbb{Z}_p, +\rangle$ is a finite group with identity 0.

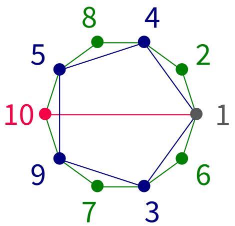
$\langle \mathbb{Z}_p^*, \cdot\rangle = \langle \mathbb{Z}_p \setminus \{0\}, \cdot\rangle$ is a finite group with identity 1 (if p prime).

Cyclic groups: Example $(\mathbb{Z}_{11}^*, \cdot)$

Consider $\mathbb{Z}_{11}^* = \{1, 2, \ldots, 10\}$ (order 10, since 11 is prime).

The subgroups of $(\mathbb{Z}_{11}^*, \cdot)$ are:

Subgroup	Generators	Order
$\{1\}$	1	1
$\{1, 10\}$	10	2
$\{1, 3, 4, 5, 9\}$	3, 4, 5, 9	5
$\{1, 2, \ldots, 10\}$	2, 6, 7, 8	10



$\mathbb{Z}_{11}^*$ is cyclic, and the elements 2, 6, 7, 8 are generators.

Reminder: Elliptic Curves

Elliptic curve over $\mathbb{F}$

Elliptic curve = solutions (x, y) of equation in Weierstrass Form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Coefficients $a_1, \ldots, a_6$ and point coordinates x, y: elements of field $\mathbb{F}$.

The Weierstrass Form can be simplified for different fields:

Elliptic curve over $\mathbb{Q}, \mathbb{R}, \mathbb{C}$, or any prime field $\mathbb{F}_{p^m}$ $(p \neq 2, 3)$

$$y^2 = x^3 + ax + b$$

(+some constraints for a, b)

Reminder: Elliptic Curves

Elliptic curve over $\mathbb{F}$

Elliptic curve = solutions (x, y) of equation in Weierstrass Form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Coefficients $a_1, \ldots, a_6$ and point coordinates x, y: elements of field $\mathbb{F}$.

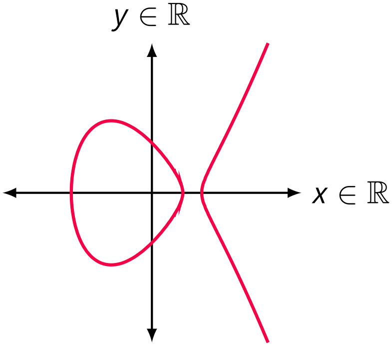
The Weierstrass Form can be simplified for different fields:

Elliptic curve over $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, or any prime field $\mathbb{F}_{p^m}$ $(p \neq 2, 3)$

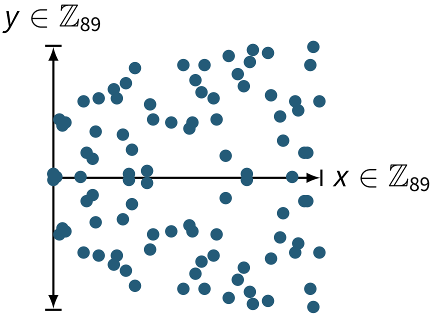
$$y^2 = x^3 + ax + b$$

(+some constraints for a, b)

Reminder: Elliptic Curves over Finite Fields



$y^2 = x^3 - 2x + 1$ over $\mathbb{R}$



$y^2 = x^3 - 2x + 1$ over $\mathbb{Z}_{89}$
(96 elements)

Reminder: The Discrete Logarithm Problem (DLP)

Definition (Discrete Logarithm Problem)

Given a prime p, a generator $g \in \mathbb{Z}_p^*$, and an element $y \in \mathbb{Z}_p^*$, find the integer $x \in \{0, \ldots, p-2\}$ such that $g^x = y \pmod{p}$.

Definition (Generalized Discrete Logarithm Problem)

Given a finite cyclic group G of order n, a generator $g \in G$ and an element $y \in G$, find $x \in \{0, \ldots, n-1\}$ such that $g^x = y$.

The difficulty of the DLP highly depends on the group.

- **Example**: DLP in $(\mathbb{Z}_p, +)$ is easy. We only need to find x such that

$$g + g + \ldots + g = g \cdot x = y \pmod{p} \quad \Rightarrow \quad x = y \cdot g^{-1} \pmod{p}.$$

- **Example**: DLP in $(\mathbb{Z}_p^*, \cdot)$ is believed to be hard.

Reminder: The Discrete Logarithm Problem (DLP)

Definition (Discrete Logarithm Problem)

Given a prime p, a generator $g \in \mathbb{Z}_p^*$, and an element $y \in \mathbb{Z}_p^*$, find the integer $x \in \{0, \ldots, p-2\}$ such that $g^x = y \pmod{p}$.

Definition (Generalized Discrete Logarithm Problem)

Given a finite cyclic group G of order n, a generator $g \in G$ and an element $y \in G$, find $x \in \{0, \ldots, n-1\}$ such that $g^x = y$.

The difficulty of the DLP highly depends on the group.

- **Example**: DLP in $(\mathbb{Z}_p, +)$ is easy. We only need to find x such that

$$g + g + \ldots + g = g \cdot x = y \pmod{p} \quad \Rightarrow \quad x = y \cdot g^{-1} \pmod{p}.$$

- **Example**: DLP in $(\mathbb{Z}_p^*, \cdot)$ is believed to be hard.

Reminder: Diffie-Hellman (DH)

Definition (Computational Diffie-Hellman Problem)

Given a finite cyclic group G of order n, a generator $g \in G$, g^a and g^b ($a, b \in \{0, \ldots, n-1\}$ and secret), find g^{ab}.

Definition (Decisional Diffie-Hellman Problem)

Given a finite cyclic group G of order n, a generator $g \in G$, distinguish the triple (g^a, g^b, g^{ab}) from (g^a, g^b, g^c). ($a, b, c \in \{0, \ldots, n-1\}$ and secret)

Best known solution: find a from g^a, or b from g^b (= solve DLP)

1-out-of-2 Oblivious Transfer

Oblivious Transfer (OT)

1-out-of-2 Oblivious Transfer: $\binom{2}{1}$-OT

- 2 parties: sender and chooser
- Sender has 2 strings $m_0, m_1 \in \{0, 1\}^n$
- Chooser has 1 bit $\nu \in \{0, 1\}$
- After protocol:
  - Chooser learns m_ν
  - Chooser learns nothing about $m_{1-\nu}$
  - Sender learns nothing about ν

Chou-Orlandi OT

- “The Simplest Protocol for Oblivious Transfer” by Chou and Orlandi [2]
- Idea:
  - Very similar to Diffie-Hellman Key-Exchange
  - Sender generates 2 encryption keys
  - Chooser is able to only learn one
- Security:
  - Based on a variant of CDH problem
  - Recent Result: Security proof has small mistake
    - needs additional fixes for malicious security

Chou-Orlandi OT (cont.)

"The Simplest Protocol for Oblivious Transfer" [2]

Alice		**Bob**
Input: (m_0, m_1)		Input: ν
$a \xleftarrow{\$} \mathbb{Z}_p$		$b \xleftarrow{\$} \mathbb{Z}_p$
	$\xrightarrow{A = g^a}$	
		if $\nu = 0$: $B = g^b$ if $\nu = 1$: $B = Ag^b$
	$\xleftarrow{B}$	
$k_0 = H(B^a)$ $k_1 = H((\frac{B}{A})^a)$		$k_R = H(A^b)$
	$\xrightarrow{e_0 \leftarrow E_{k_0}(m_0),\ e_1 \leftarrow E_{k_1}(m_1)}$	
		$m_\nu = D_{k_R}(e_\nu)$

Naor-Pinkas OT

- Efficient OT protocol by Naor and Pinkas [4]
- Idea:
  - Sender generates 2 encryption keys
  - Chooser is able to only learn one
- Security:
  - Based on DDH problem
  - Secure for groups with prime order q, where DDH is hard

Naor-Pinkas OT (cont.)

Naor-Pinkas Oblivious Transfer [4]

Alice		**Bob**
Input: (m_0, m_1)	Public parameters: mult. Group G of prime order q, generator g	Input: ν
		$a, b \xleftarrow{\$} \mathbb{Z}_q$
		$c_\nu = ab, c_{1-\nu} \xleftarrow{\$} \mathbb{Z}_q$
		$x = g^a, y = g^b$
		$z_0 = g^{c_0}, z_1 = g^{c_1}$
	$\xleftarrow{(x, y, z_0, z_1)}$	
verify $z_0 \neq z_1$		
$r_0, s_0, r_1, s_1 \xleftarrow{\$} \mathbb{Z}_q$		
$w_0 = x^{s_0} \cdot g^{r_0}, k_0 = z_0^{s_0} \cdot y^{r_0}$		
$w_1 = x^{s_1} \cdot g^{r_1}, k_1 = z_1^{s_1} \cdot y^{r_1}$		
	$\xrightarrow{(w_0, \mathbb{E}_{k_0}(m_0), w_1, \mathbb{E}_{k_1}(m_1))}$	
		$k_\nu = (w_\nu)^b$
		$m_\nu = \mathbb{E}_{k_\nu}^{-1}(\mathbb{E}_{k_\nu}(m_\nu))$

1-out-of-N Oblivious Transfer

Oblivious Transfer (cont.)

1-out-of-N Oblivious Transfer: $\binom{N}{1}$-OT

- 2 parties: sender and chooser
- Sender has N strings $m_0, \ldots, m_{N-1} \in \{0,1\}^n$
- Chooser has $\log_2(N)$-bit value $\nu \in \{0, \ldots, N-1\}$
- After protocol:
  - Chooser learns m_ν
  - Chooser learns nothing about $m_i \neq m_\nu$
  - Sender learns nothing about ν

Building $\binom{N}{1}$-OT

- Instantiate directly if supported (e.g., Naor-Pinkas OT)
- Build from $\binom{2}{1}$-OT (Idea: transfer encryption key per bit of ν)

$\binom{N}{1}$-OT from $\binom{2}{1}$-OT

- Sender prepares $L = \log_2(N)$ keypairs: $(k_1^0, k_1^1), \ldots, (k_L^0, k_L^1)$
- Sender encrypts and sends item m_i:

$$C_i = m_i \oplus \left(\bigoplus_{j=1}^{L} \mathbb{E}(k_j^{i_j}, i) \right)$$

- For each bit b of ν: Perform $\binom{2}{1}$-OT to give chooser k_j^b
- Chooser has all keys to decrypt C_ν

Solution to Millionaires' Problem

- We can use $\binom{N}{1}$-OT to solve the Millionaires' Problem
  - Constraint: Set of possible inputs is small
  - Assume Alice and Bob have $i \in S = \{1, \ldots, 10\}$ million
- Protocol:
  - Alice has 5 million, Bob has 3 million.
  - Alice calculates $F(5, y)$ for all $y \in S$
    - $F(5, \{1, \ldots, 4\})$ = 'Alice'
    - $F(5, 5)$ = 'Same'
    - $F(5, \{6, \ldots, 10\})$ = 'Bob'

Solution to Millionaires' Problem

- We can use $\binom{N}{1}$-OT to solve the Millionaires' Problem
  - Constraint: Set of possible inputs is small
  - Assume Alice and Bob have $i \in S = \{1, \ldots, 10\}$ million
- Protocol:
  - Alice has 5 million, Bob has 3 million.
  - Alice calculates $F(5, y)$ for all $y \in S$
    - $F(5, \{1, \ldots, 4\})$ = 'Alice'
    - $F(5, 5)$ = 'Same'
    - $F(5, \{6, \ldots, 10\})$ = 'Bob'

Solution to Millionaires' Problem (cont.)

- Protocol (cont.)
  - Perform $\binom{10}{1}$-OT
  - Alice inputs the 10 results: {'Alice',…,'Same',…,'Bob}
  - Bob's choice is his input value: $\nu = 3$
  - After the OT, he learns 'Alice' and tells the result to Alice
- Problems:
  - Only practical for small input sets
    - More on that now
  - Only works for honest parties, cannot detect cheating
    - More on that next lecture

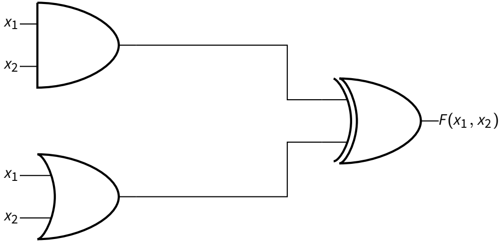
Solution to Millionaires' Problem (cont.)

- Protocol (cont.)
  - Perform $\binom{10}{1}$-OT
  - Alice inputs the 10 results: {'Alice',...,'Same',...,'Bob}
  - Bob's choice is his input value: $\nu = 3$
  - After the OT, he learns 'Alice' and tells the result to Alice
- Problems:
  - Only practical for small input sets
    - More on that now
  - Only works for honest parties, cannot detect cheating
    - More on that next lecture

Protocols for Multiparty Computation

Multiparty Computation (MPC) Protocols

- Shared Idea:
  - Jointly evaluate a circuit calculating $F(x_1, x_2)$
- Yao's garbled circuits [5]
- Goldreich-Micali-Widgerson protocol [3]



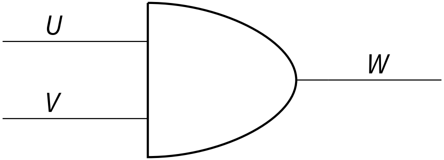
Yao's Garbled Circuits

Basics

- Yao's original solution to his Millionaires' Problem
- Basic Idea:
  - Describe function as boolean circuit
  - Obfuscate (garble) truth tables of gates in circuit
  - Encrypt output values with corresponding input values
    - Allows to decrypt only 1 output
    - No idea if current wire is 0 or 1 due to garbling

Example: Garbling an AND Gate

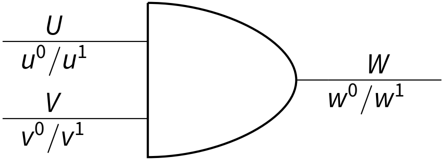
- Draw truth table of gate



U	V	W
0	0	0
0	1	0
1	0	0
1	1	1

Example: Garbling an AND Gate

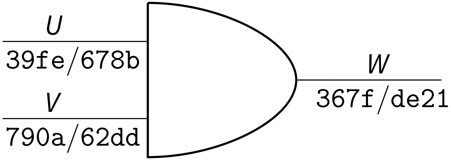
- Assign labels to each wire



U	V	W
0	0	0
0	1	0
1	0	0
1	1	1

Example: Garbling an AND Gate

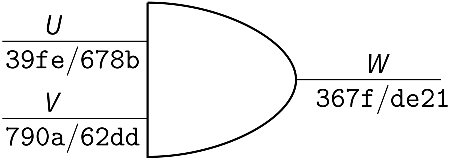
- Replace boolean values with wire labels



U	V	W
39fe	790a	367f
39fe	62dd	367f
678b	790a	367f
678b	62dd	de21

Example: Garbling an AND Gate

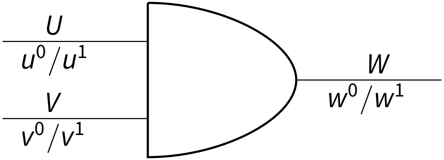
- Encrypt result of gate with input labels



U	V	W
39fe	790a	$\mathbb{E}_{\texttt{39fe},\texttt{790a}}(\texttt{367f})$
39fe	62dd	$\mathbb{E}_{\texttt{39fe},\texttt{62dd}}(\texttt{367f})$
678b	790a	$\mathbb{E}_{\texttt{678b},\texttt{790a}}(\texttt{367f})$
678b	62dd	$\mathbb{E}_{\texttt{678b},\texttt{62dd}}(\texttt{de21})$

Example: Garbling an AND Gate

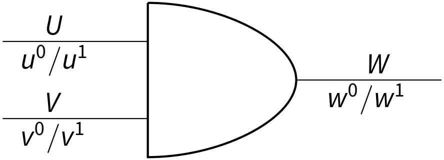
- Generalized view



U	V	W
u^0	v^0	$\mathbb{E}_{u^0,v^0}(w^0)$
u^0	v^1	$\mathbb{E}_{u^0,v^1}(w^0)$
u^1	v^0	$\mathbb{E}_{u^1,v^0}(w^0)$
u^1	v^1	$\mathbb{E}_{u^1,v^1}(w^1)$

Example: Garbling an AND Gate

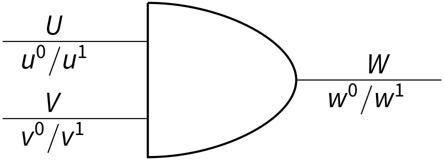
- Randomly shuffle table rows



U	V	W
u^1	v^0	$\mathbb{E}_{u^1,v^0}(w^0)$
u^1	v^1	$\mathbb{E}_{u^1,v^1}(w^1)$
u^0	v^1	$\mathbb{E}_{u^0,v^1}(w^0)$
u^0	v^0	$\mathbb{E}_{u^0,v^0}(w^0)$

Example: Garbling an AND Gate

- Only keep encrypted result



Garbled Table
$\mathbb{E}_{u^1,v^0}(w^0)$
$\mathbb{E}_{u^1,v^1}(w^1)$
$\mathbb{E}_{u^0,v^1}(w^0)$
$\mathbb{E}_{u^0,v^0}(w^0)$

Garbling a Circuit

- Assign wire labels for 0 and 1 to each wire
- Garble gate:
  - Replace truth table by corresponding wire labels
  - Encrypt output wire labels using input wire labels as keys
  - Randomly shuffle entries in truth table
- Repeat for each gate

Input Values

- Alice is the garbler
  - She knows the wire labels corresponding to her input
  - Can send her wire labels and circuit to Bob
- How does Bob get the wire labels corresponding to his input?
  - Cannot tell Alice his input values directly
- Solution: Use $\binom{2}{1}$-OT!
  - For each input wire w_i corresponding to Bob's input bit y_i
  - Alice is OT-sender with strings w_i^0, w_i^1
  - Bob is OT-chooser with choice bit y_i

Input Values

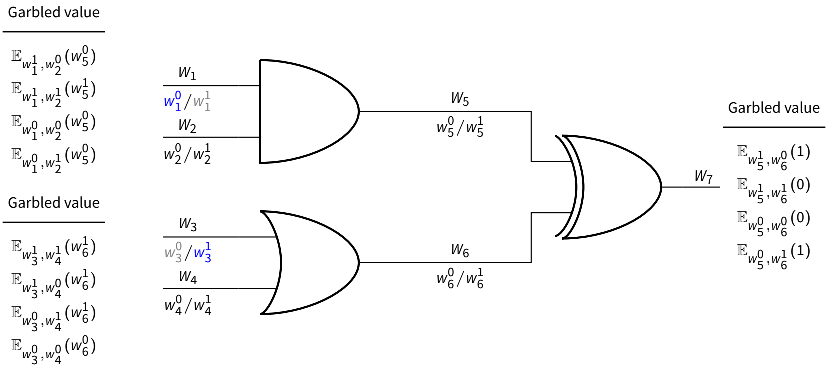
- Alice is the garbler
  - She knows the wire labels corresponding to her input
  - Can send her wire labels and circuit to Bob
- How does Bob get the wire labels corresponding to his input?
  - Cannot tell Alice his input values directly
- Solution: Use $\binom{2}{1}$-OT!
  - For each input wire w_i corresponding to Bob's input bit y_i
  - Alice is OT-sender with strings w_i^0, w_i^1
  - Bob is OT-chooser with choice bit y_i

Output Values

- After the evaluation Bob only has the output wire label o^x
- Alice is the garbler, and therefore knows the corresponding value
  - Communicate so one or both parties learn the output
- Other possibility:
  - Do not assign wire labels to output values
  - Last garbled table decrypts directly to 0 or 1

Example Evaluation

- Bob receives the garbled circuit and Alice's input labels

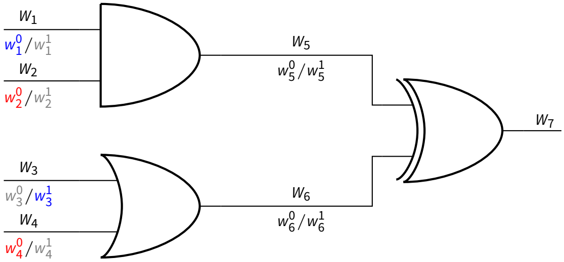


Example Evaluation

- Bob uses $\binom{2}{1}$-OT to receive the wire labels for his input

Garbled value
$\mathbb{E}_{w_1^1, w_2^0}(w_5^0)$
$\mathbb{E}_{w_1^1, w_2^1}(w_5^1)$
$\mathbb{E}_{w_1^0, w_2^0}(w_5^0)$
$\mathbb{E}_{w_1^0, w_2^1}(w_5^0)$

Garbled value
$\mathbb{E}_{w_3^1, w_4^1}(w_6^1)$
$\mathbb{E}_{w_3^1, w_4^0}(w_6^1)$
$\mathbb{E}_{w_3^0, w_4^1}(w_6^1)$
$\mathbb{E}_{w_3^0, w_4^0}(w_6^0)$



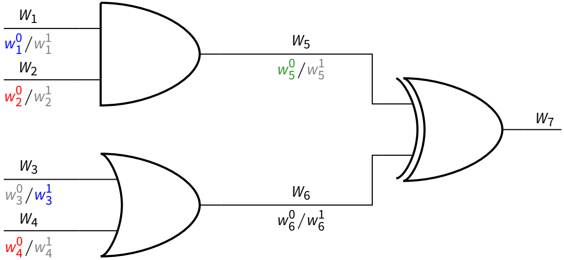
Garbled value
$\mathbb{E}_{w_5^1, w_6^0}(1)$
$\mathbb{E}_{w_5^1, w_6^1}(0)$
$\mathbb{E}_{w_5^0, w_6^0}(0)$
$\mathbb{E}_{w_5^0, w_6^1}(1)$

Example Evaluation

- Bob evaluates the AND gate

Garbled value
$\mathbb{E}_{w_1^1, w_2^0}(w_5^0)$
$\mathbb{E}_{w_1^1, w_2^1}(w_5^1)$
$\mathbb{E}_{w_1^0, w_2^0}(w_5^0)$
$\mathbb{E}_{w_1^0, w_2^1}(w_5^0)$

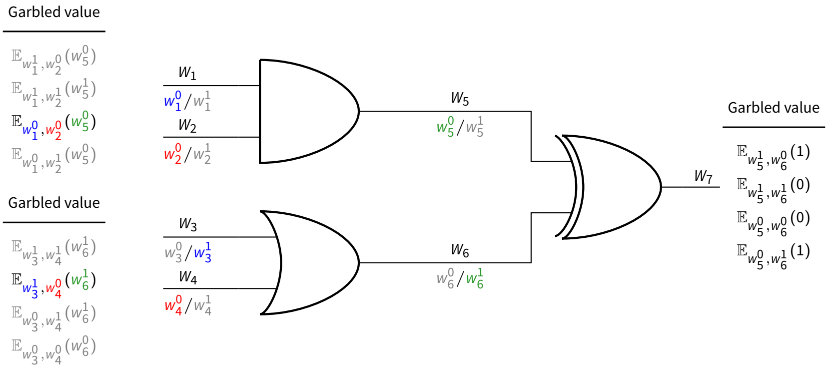
Garbled value
$\mathbb{E}_{w_3^1, w_4^1}(w_6^1)$
$\mathbb{E}_{w_3^1, w_4^0}(w_6^1)$
$\mathbb{E}_{w_3^0, w_4^1}(w_6^1)$
$\mathbb{E}_{w_3^0, w_4^0}(w_6^0)$



Garbled value
$\mathbb{E}_{w_5^1, w_6^0}(1)$
$\mathbb{E}_{w_5^1, w_6^1}(0)$
$\mathbb{E}_{w_5^0, w_6^0}(0)$
$\mathbb{E}_{w_5^0, w_6^1}(1)$

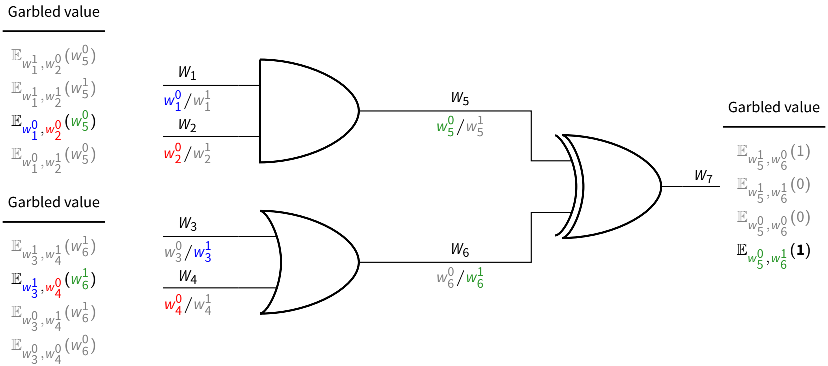
Example Evaluation

- Bob evaluates the OR gate



Example Evaluation

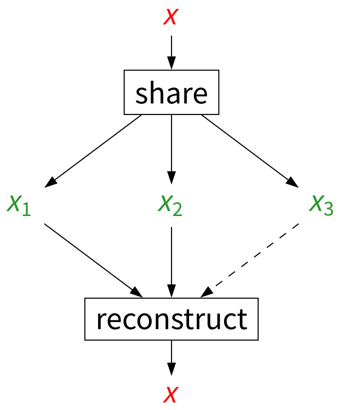
- Bob evaluates the XOR gate and obtains the result



Goldreich-Micali-Wigderson

Recall: Secret Sharing

- Split a secret x into shares
- Each party gets a share
- Individual shares give no information about x
- Parties combine their shares to reconstruct x
- Different instantiations of “share” and “reconstruct”
  - n-out-of-n: all n shares are needed to reconstruct
  - k-out-of-n: any k shares suffice to reconstruct



Recall: Additive Secret Sharing

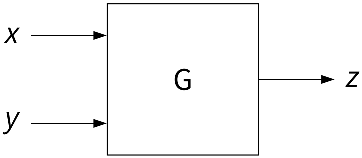
- For an arbitrary group $\langle G, \circ \rangle$ we can share any $x \in G$:
  - Pick $x_A \in G$ at random
  - Define $x_B = {x_A}^{-1} \circ x$
  - Given only one of x_A or x_B, x is perfectly hidden
  - Given both, x can be restored ($x = x_A \circ x_B$)
  - Extends to arbitrary amount of shares by picking all but last share as random
- Example group $\langle \{0, 1\}, \oplus \rangle$:
  - Shares of x are $x_A \xleftarrow{\$} \{0, 1\}$ and $x_B = x \oplus x_A$
  - Reconstruct: $x = x_A \oplus x_B$

Recall: Additive Secret Sharing

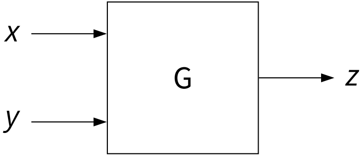
- For an arbitrary group $\langle G, \circ \rangle$ we can share any $x \in G$:
  - Pick $x_A \in G$ at random
  - Define $x_B = {x_A}^{-1} \circ x$
  - Given only one of x_A or x_B, x is perfectly hidden
  - Given both, x can be restored ($x = x_A \circ x_B$)
  - Extends to arbitrary amount of shares by picking all but last share as random
- Example group $\langle \{0, 1\}, \oplus \rangle$:
  - Shares of x are $x_A \stackrel{\$}{\leftarrow} \{0, 1\}$ and $x_B = x \oplus x_A$
  - Reconstruct: $x = x_A \oplus x_B$

GMW Protocol

- By Goldreich, Micali and Wigderson (1987)
- Basic Idea:
  - Secret-share all input with other party
  - Build logic circuit with 2-input gates (e.g., AND, XOR)
  - Jointly evaluate each gate ($z = G(x, y)$)



GMW (cont.)



- Additive Secret Sharing using XOR
- Alice knows x_A, y_A, Bob knows x_B, y_B
- Alice samples z_A at random

$$z_B = z_A \oplus G(x_A \oplus x_B, y_A \oplus y_B)$$

GMW (cont.)

$$z_B = z_A \oplus G(x_A \oplus x_B, y_A \oplus y_B)$$

- Alice can use $\binom{4}{1}$-OT to give Bob his share of the result
- Due to properties of OT, Alice learns nothing about x_B, y_B
- Due to properties of OT and secret sharing, Bob learns nothing about x_A, y_A
- Let's demonstrate for an AND gate:

x_B	y_B	ν	z_B
0	0	0	$((x_A \oplus 0) \& (y_A \oplus 0)) \oplus z_A$
0	1	1	$((x_A \oplus 0) \& (y_A \oplus 1)) \oplus z_A$
1	0	2	$((x_A \oplus 1) \& (y_A \oplus 0)) \oplus z_A$
1	1	3	$((x_A \oplus 1) \& (y_A \oplus 1)) \oplus z_A$

GMW (cont.)

- We can now efficiently handle one gate G
- Whole Circuit?
  - Each party secret-shares its input values with other party
  - Agree on order of the gates in the circuit
  - Alice evaluates each gate, acting as the OT-sender
  - Bob acts as the OT-chooser and gets his share of the output
  - Iteratively compute the whole circuit
  - Exchange shares of output gate(s) at the end

GMW (cont.)

- Bonus: XOR gates do not require OT
  - Additive Secret Sharing!
  - Set $z_A = x_A \oplus y_A,\ z_B = x_B \oplus y_B$
- Proof:

$$\begin{aligned} z = z_A \oplus z_B &= (x_A \oplus y_A) \oplus (x_B \oplus y_B) \\ &= (x_A \oplus x_B) \oplus (y_A \oplus y_B) = x \oplus y. \end{aligned}$$

Summary

Summary

- MPC Protocols
  - Yao's Garbled Circuits [5]
  - GMW [3]
- Powered by OT
- Solution to Millionaires' Problem
  - Build boolean circuit to evaluate $x < y$
  - Use Yao or GMW to evaluate

Questions you should be able to answer

1. What is oblivious transfer and what are the properties for the sender and receiver? Give an example for an OT protocol.
2. Describe the steps involved Yao's garbled circuit protocol. When do the parties need to interact with each other?
3. Describe the GMW protocol. When do the parties need to interact with each other? What needs to be done during this interaction?
4. Alice and Bob want to know which of them has more money, without disclosing their respective amounts. Give a detailed solution to this problem using either GMW or Yao.

Bibliography I

[1] Mihir Bellare and Silvio Micali. **Non-Interactive Oblivious Transfer and applications.** CRYPTO. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 547–557.

[2] Tung Chou and Claudio Orlandi. **The Simplest Protocol for Oblivious Transfer.** LATINCRYPT. Vol. 9230. Lecture Notes in Computer Science. Springer, 2015, pp. 40–58.

[3] Oded Goldreich, Silvio Micali, and Avi Wigderson. **How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority.** STOC. ACM, 1987, pp. 218–229.

[4] Moni Naor and Benny Pinkas. **Efficient oblivious transfer protocols.** SODA. ACM/SIAM, 2001, pp. 448–457.

[5] Andrew Chi-Chih Yao. **How to Generate and Exchange Secrets (Extended Abstract).** FOCS. IEEE Computer Society, 1986, pp. 162–167.

[6] Andrew Chi-Chih Yao. **Protocols for Secure Computations (Extended Abstract).** FOCS. IEEE Computer Society, 1982, pp. 160–164.