Secure Computation - III

CS 601.642/442 Modern Cryptography

Fall 2019

Securely Computing *any* Function

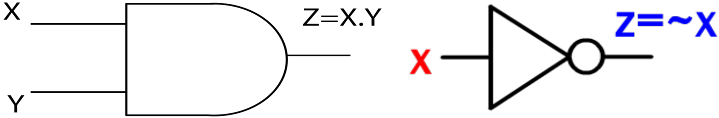
How can a group of parties securely compute *any* function over their private inputs?

- **Last time:** Yao's Garbled Circuits based solution. Requires little interaction, but only tailored to two-party case.
- **Today:** Goldreich-Micali-Wigderson (GMW) solution. Highly interactive. But extends naturally to $n > 2$ parties (where up to $n - 1$ parties may be corrupted).

Circuit Representation

Function $f(x, y)$ can be written as a boolean circuit C:

- *Input*: Input wires of C correspond to inputs x and y to f
- *Gates*: C contains AND and NOT gates, where each gate has fan in at most 2 and arbitrary fan out



- *Output*: Output wires of C correspond to output of $f(x, y)$

Secret Sharing

A k-out-of-n secret sharing scheme allows for "dividing" a secret value s into n parts $s_1, \ldots, s_n$ s.t.

- **Correctness:** Any subset of k shares can be "combined" to reconstruct the secret s
- **Privacy:** The value s is completely hidden from anyone who only has at most $k-1$ shares of s

Think: How to formalize?

Secret Sharing: Definition

Definition

A (k, n) secret-sharing consists of a pair of PPT algorithms $(\mathsf{Share}, \mathsf{Reconstruct})$ s.t.:

- $\mathsf{Share}(s)$ produces an n tuple $(s_1, \ldots, s_n)$
- $\mathsf{Reconstruct}(s'_{i_1}, \ldots, s'_{i_k})$ is s.t. if $\{s'_{i_1}, \ldots, s'_{i_k}\} \subseteq \{s_1, \ldots, s_n\}$, then it outputs s
- For any two s and $\tilde{s}$, and for any subset of at most $k-1$ indices $X \subset [1, n]$, $|X| < k$, the following two distributions are statistically close:

$$\left\{(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s) : (s_i | i \in X)\right\},$$

$$\left\{(\tilde{s}_1, \ldots, \tilde{s}_n) \leftarrow \mathsf{Share}(\tilde{s}) : (\tilde{s}_i | i \in X)\right\}.$$

Secret Sharing: Construction

An (n, n) secret-sharing scheme for $s \in \{0, 1\}$ based on XOR:

- $\mathsf{Share}(s)$: Sample random bits $(s_1, \ldots, s_n)$ s.t. $s_1 \oplus \cdots \oplus s_n = s$
- $\mathsf{Reconstruct}(s'_1, \ldots, s'_n)$: Output $s'_1 \oplus \cdots \oplus s'_n$

Think: Security?

Additional Reading: Shamir's (k, n) secret-sharing using polynomials

GMW Protocol: Outline

GMW protocol consists of three phases:

- **Input Sharing:** Each party *secret-shares* its input into two parts and sends one part to the other party
- **Circuit evaluation:** The parties evaluate the circuit in a *gate-by-gate* fashion in such a manner that for every internal wire w in the circuit, each party holds a secret share of the value of wire w
- **Output reconstruction:** Finally, the parties exchange the secret shares of the output wires. Each party then, on its own, combines the secret shares to compute the output of the circuit

GMW Protocol: Details

Notation:

- **Protocol Ingredients:** A $(2,2)$ secret-sharing scheme $(\mathsf{Share}, \mathsf{Reconstruct})$, and a 1-out-of-4 OT scheme $(\mathsf{OT} = (S, R))$
- **Common input:** Circuit C for function $f(\cdot,\cdot)$ with two n-bit inputs and an n-bit output
- **A's input:** $x = x_1, \ldots, x_n$ where $x_i \in \{0,1\}$
- **B's input:** $y = y_1, \ldots, y_n$ where $y_i \in \{0,1\}$

Protocol Invariant: For every wire in $C(x,y)$ with value $w \in \{0,1\}$, A and B have shares w^A and w^B, respectively, s.t. $\mathsf{Reconstruct}(w^A, w^B) = w$

GMW Protocol: Details (contd.)

Protocol $\Pi = (A, B)$**:**

Input Sharing: A computes $(x_i^A, x_i^B) \leftarrow \mathsf{Share}(x_i)$ for every $i \in [n]$ and sends $(x_1^B, \ldots, x_n^B)$ to B. B acts analogously.

Circuit Evaluation: Run the $\mathsf{CircuitEval}$ sub-protocol. A obtains out_i^A and B obtains out_i^B for every output wire i.

Output Phase: For every output wire i, A sends out_i^A to B, and B sends out_i^B to A. Each party computes

$$\mathsf{out}_i = \mathsf{Reconstruct}(\mathsf{out}_i^A, \mathsf{out}_i^B)$$

The output is $\mathsf{out} = \mathsf{out}_1, \ldots, \mathsf{out}_n$

CircuitEval: NOT Gate

NOT Gate: Input u, output w

- A holds u^A, B holds u^B
- A computes $w^A = u^A \oplus 1$
- B computes $w^B = u^B$

Observe: $w^A \oplus w^B = u^A \oplus 1 \oplus u^B = \bar{u}$

CircuitEval: AND Gate

AND Gate: Inputs u, v, output w

- A holds u^A, v^A, B holds u^B, v^B
- A samples $w^A \xleftarrow{\$} \{0,1\}$ and computes $w_1^B, \ldots, w_4^B$ as follows:

u^B	v^B	w^B
0	0	$w_1^B = w^A \oplus \left((u^A \oplus 0) \cdot (v^A \oplus 0)\right)$
0	1	$w_2^B = w^A \oplus \left((u^A \oplus 0) \cdot (v^A \oplus 1)\right)$
1	0	$w_3^B = w^A \oplus \left((u^A \oplus 1) \cdot (v^A \oplus 0)\right)$
1	1	$w_4^B = w^A \oplus \left((u^A \oplus 1) \cdot (v^A \oplus 1)\right)$

- A and B run $\mathsf{OT} = (S, R)$ where A acts as sender S with inputs $(w_1^B, \ldots, w_4^B)$ and B acts as receiver R with input $b = 1 + 2u^B + v^B$

Intuition for Security

For every wire in C (except the input and output wires), each party only holds a secret share of the wire value:

- **NOT gate:** Follows from construction
- **AND gate:** Follows from security of OT

At the end, the parties only learn the values of the output wires

<u>Exercise</u>: Construct Simulator for Π using Simulator for OT and prove indistinguishability