1 GMW Protocol, cont’d

Recall that the GMW protocol for securely computing any circuit proceeds in three phases:

1. Input sharing
2. Circuit evaluation (using 1-out-of-4 OT)
3. Output recovery

1.1 Security (in the semi-honest OT-hybrid model)

OT-hybrid model means we have ideal functionalities computing the 1-out-of-4 OT’s. Security here is information theoretic. To show security, we construct the following two simulators that simulate respectively each player’s view. A simulator of P₁’s view is indistinguishable from P₁’s view during a correct protocol execution, however it is defined independently of P₂’s private inputs, therefore demonstrating that the protocol leaks no information. Likewise for P₂’s view.

1.1.1 Sim₁(\(x₁, y₁\)):

- 1a) generate random values \(r₁, \ldots, r_\ell\) (P₁’s randomness)
- 1b) generate random values \(s₁, \ldots, s_\ell\) (initial message)
- 2) for each non-input wire of the circuit, generate random \(r_i\)
- 3) for each output wire of \(P₁\) let \(\hat{r}_1, \ldots, \hat{r}_\ell\) be \(P₁\)’s shares of the wires
  - let \(sh_i = \hat{r}_i \oplus y₁, i\)
  - place \(sh₁, \ldots, sh_\ell\) in the view

1.1.2 Sim₂(\(x₂, y₂\)):

- 1) same as Sim₁
- 2) for each non-input wire of the circuit, choose uniform \(r_i\) as the output of the functionality
- 3) same

Note: For Sim₁, \(x₁\) is \(P₁\)’s input, while \(y₁\) is player \(P₂\)’s output. However, for Sim₂, \(x₂\) is \(P₁\)’s output, and \(y₂\) is \(P₂\)’s input.
1.2 Randomized Computations

If we can securely compute any deterministic function, then we can securely compute any randomized functionality. Let $g(x, y)$ be a randomized function; we can construct a deterministic equivalent $\hat{g}(x, y, r) = g(x, y)$ where the random coins $r$ are chosen uniformly at random. Then we can securely compute $\overline{g}((x, r_1), (y, r_2)) = \hat{g}(x, y, r_1 \oplus r_2)$ using GMW.

- 1) Each player $P_i$ chooses uniform $r_i$
- 2) parties compute $\overline{g}$ on inputs $(x, r_1)$ and $(y, r_2)$ respectively

2 Yao’s Garbled-Circuits

Motivation: GMW protocol has round complexity linear in the depth of the circuit. Yao’s garbled-circuit approach has $O(1)$ round complexity, with a pretty small constant.

One party acts as a garbled circuit generator. For each wire, she generates a pair of symmetric encryption keys, corresponding to a possible value (0 or 1). For each gate (assume two input wires, one output wire), she constructs a garbled table representing the truth table for the gate. (Note: the table should be randomly permuted). The following example is for an AND gate:

$$
\begin{array}{c|c|c|c|c|c}
\text{label}_1 & \text{label}_2 & \text{garbled key} \\
0 & 0 & \text{Enc}_{k_0}(\text{Enc}_{k_0'}(k_0'')) \\
0 & 1 & \text{Enc}_{k_0}(\text{Enc}_{k_0'}(k_0'')) \\
1 & 0 & \text{Enc}_{k_1}(\text{Enc}_{k_0}(k_0'')) \\
1 & 1 & \text{Enc}_{k_1}(\text{Enc}_{k_0'}(k_0'')) \\
\end{array}
$$

Instead of having to decrypt every row, evaluation can be simplified if the garbled circuit generator also chooses a random bit $\lambda$. Thus the label of $k_0$ will be $\lambda \oplus b$. Then the circuit evaluator can use the label to immediately access the correct row of the table.
2.1 Garbled-circuit protocol

- 1) Input-preparation phase
  - 1a) $P_1$ sends the input-wire keys corresponding to his inputs
  - 1b) $P_2$ obtains the input-wire keys for its inputs using 1-out-of-2 OT
- 2) Garbled-circuit evaluation
- 3) Output determination:
  - $P_1$ sends $\lambda$-values on output wires, $P_2$ sends output keys
  - $P_1$ sends $\lambda$-values on output wires, $P_2$ sends output keys