CS 507, Topics in Cryptography: Secure Computation
Homework 3

Due: November 14, 2025

Problem 1. Recall that the basic Garbled Circuit technique does not offer protection against a malicious garbler. In this problem, you will construct attacks by a malicious garbler.

Suppose we consider a basic version of Yao's GC that encodes circuits as follows. First, the (honest) garbler assigns to each wire w (1) two uniform keys $k_w^0, k_w^1 \in_\$ \{0,1\}^\lambda$ and (2) a uniform color bit (for point and permute) $c_w \in_\$ \{0,1\}$. Let $\texttt{Enc}$ denote a CPA\$-secure encryption algorithm. I.e., each encryption looks (to a party that does not hold the encryption key) like a uniform string. The (honest) garbler constructs the four-row garbling $(r_{00}, r_{01}, r_{10}, r_{11})$ of each AND gate $z \leftarrow x \cdot y$ as follows:

$$
\begin{aligned}
&\text{let } (k_x^0, k_x^1), (k_y^0, k_y^1), (k_z^0, k_z^1) \text{ be keys on } x, y, z, \text{ respectively} \\
&\text{let } \alpha, \beta, \gamma \text{ be colors on } x, y, z, \text{ respectively} \\
&r_{00} \leftarrow \texttt{Enc}(k_x^\alpha, \texttt{Enc}(k_y^\beta, (k_z^{\alpha\beta}, \alpha\beta \oplus \gamma))) \\
&r_{01} \leftarrow \texttt{Enc}(k_x^\alpha, \texttt{Enc}(k_y^{\bar\beta}, (k_z^{\alpha\bar\beta}, \alpha\bar\beta \oplus \gamma))) \\
&r_{10} \leftarrow \texttt{Enc}(k_x^{\bar\alpha}, \texttt{Enc}(k_y^\beta, (k_z^{\bar\alpha\beta}, \bar\alpha\beta \oplus \gamma))) \\
&r_{11} \leftarrow \texttt{Enc}(k_x^{\bar\alpha}, \texttt{Enc}(k_y^{\bar\beta}, (k_z^{\bar\alpha\bar\beta}, \bar\alpha\bar\beta \oplus \gamma)))
\end{aligned}
$$

The main invariant on each wire w is as follows: Let x be the true value on that wire and let α be that wire's color. The evaluator obtains (1) the key k_w^x corresponding to the true value and (2) the masked value $x \oplus \alpha$. The evaluator evaluates AND gates by using her input keys to decrypt the row corresponding to her input wire color bits.

Now, consider a circuit C that takes as input (1) a length-10 string $x \in \{0,1\}^{10}$ from the garbler and (2) a length-10 string $y \in \{0,1\}^{10}$ from the evaluator and that outputs a single bit, which is the AND of all inputs:

$$C(x, y) = \left(\prod_i x_i \cdot y_i \right)$$

Consider the following scenario:

- The garbler garbles the circuit: $\hat{C}, X \leftarrow \text{Garble}(C)$. Here, $\hat{C}$ is the garbled circuit and X is a vector of pairs of keys for the circuit input.
- The evaluator obtains (via a TTP) input keys from X that correspond to her input and the garbler's inputs.

- The garbler sends to the evaluator $\hat{C}$.
- The evaluator evaluates the garbled circuit and obtains a single length-λ key on the single output wire.
- The evaluator sends this output key to the garbler.

1. Describe a malicious adversary $\mathcal{A}$ that can in this scenario learn some single bit y_j of the evaluator's input, for its choice of j. Namely, your adversary does not call Garble as prescribed, but instead runs arbitrary code to build a "garbled circuit". Note that your "garbled circuit" should be indistinguishable from an honestly constructed one such that the evaluator cannot detect that the attack is happening.
2. Now, describe a new adversary $\mathcal{A}$ that can with overwhelming probability learn the evaluator's *entire input*.

   Hint: You may assume that the security parameter $\lambda \gg 10$.

Answer 1.

Problem 2 (Generalizing GMW)**.** In the context of this problem, restrict consideration to two-party computation.

In lecture, we described the GMW protocol as a technique for securely computing fan-in two Boolean circuits. We gave a reduction from semi-honest 2PC to the following preprocessing functionality:

- The parties A and B may invoke this functionality $\mathcal{F}$ as many times as they would like.
- $\mathcal{F}$ samples a Beaver triple:

$$\{\ \alpha, \beta, \alpha \cdot \beta \ \text{ where } \ \alpha, \beta \in_{\$} \{0,1\}\ \}$$

- $\mathcal{F}$ samples uniform XOR sharings $[\alpha], [\beta], [\alpha \cdot \beta]$ and gives one share to each party.

Based on this functionality, we showed how to securely evaluate any Boolean circuit consisting of fan-in-two AND and XOR gates. It is interesting to consider richer—and therefore potentially more efficient—circuit representations.

1. **Arithmetic GMW.** In this part, you will generalize the GMW protocol to work with integer values[1] instead of bits. Suppose we wish to compute an *arithmetic* circuit where each wire holds an integer modulo $m \in \mathbb{N}$, rather than a bit. In particular, each gate in the circuit is now an addition/multiplication gate over $\mathbb{Z}_m$, rather than XOR/AND.
   (a) Describe (1) a modification of the above preprocessing functionality and (2) how the parties use your modified functionality $\mathcal{F}'$ to handle each gate type.
   (b) Prove your protocol is *correct*.
   (c) Prove your protocol is semi-honest secure in the $\mathcal{F}'$-hybrid model.

[1] It is also possible to generalize GMW to work with arbitrary *finite rings*, though you do not need to do this here.

2. **Boolean GMW with Vector-Scalar Multiplication.** In this part, you will extend Boolean GMW with a richer gate type. In particular, you will generate appropriate preprocessing needed to support a *vector-scalar multiplication gate*. Such a gate takes as input (1) a vector $x \in \{0,1\}^n$ and (2) a scalar s. It outputs the scaled vector $s \cdot x$. Note that such a gate generalizes the notion of an AND gate. Consider the following modified preprocessing functionality:

   - The parties A and B may invoke this functionality $\mathcal{F}'$ as many times as they would like.
   - Parties agree on a vector length n, and $\mathcal{F}'$ samples a vector-scalar triple:

   $$\{\ \alpha, \beta, \alpha \cdot \beta \ \text{ where } \ \alpha \in_\$ \{0,1\}, \beta \in_\$ \{0,1\}^n\ \}$$

   - $\mathcal{F}'$ samples uniform XOR sharings $[\alpha], [\beta], [\alpha \cdot \beta]$ and gives one share to each party.

   (a) Describe a protocol $\Pi_{\texttt{pre}}$ that uses 1-out-of-2 string OT to instantiate $\mathcal{F}'$. *Your number of consumed OTs per triple should be independent of n.*

   (b) Prove your preprocessing protocol is correct.

   (c) Prove your preprocessing protocol is semi-honest secure in the OT-hybrid model.

Answer 2.

Problem 3 (Limitations of Circuits)**.** In this course, we focused our discussion on Boolean circuits, and we constructed several powerful techniques for securely implementing Boolean circuits. Circuits form a complete model of computation, and some computations with circuits are efficient, but some are not. Consider the following simple functionality that accesses an element from a list:

PARAMETERS:

1. Let P_0, P_1 be two parties.
2. Let n denote a list size, and let $n = 2^k$ for some k. I.e., $k = \log_2 n$.

FUNCTIONALITY:

1. P_0 inputs a list of n bits $X \in \{0,1\}^n$.
2. P_1 inputs an index $i \in \{0,1\}^k$.
3. Party P_1 outputs $X[i]$, where i is interpreted as an integer index.
4. Party P_0 outputs $\perp$.

1. **Base protocol.** First, suppose that we do not care about security (i.e., suppose each party is honest). Describe a simple protocol that implements the above functionality in $O(k)$ time.

2. Now, suppose we care about protecting (1) P_0's unaccessed bits, and (2) P_1's index. I.e., we wish to implement the functionality with a semi-honest protocol. We decide to implement the functionality via GMW or GC, and so we must prepare a Boolean circuit that implements the functionality. Design a Boolean circuit with $O(n)$ gates that implements array lookup; i.e., on input (X, i) the circuit outputs $X[i]$.

3. Is it possible to build a circuit that (1) correctly solves the same problem and (2) has size (i.e., number of gates) that scales only sublinearly with n (i.e., has $o(n)$ gates)? Why/why not?

Answer 3.