Yaos – Homework

Please answer the following questions. **We don't expect rigorous formal proofs: rather, just a high-level argument from intuition.** Please submit your answers as a **PDF** to Gradescope. Collaboration is allowed and encouraged, but you must write up your own answers and acknowledge your collaborators in your submission.

Due Date: *Monday, April 21st*

1 Crypto Notions

(1) In one sentence, what is secure two-party/multi-party computation?

(2) Give a potential application of secure 2PC/MPC in practice. (Try to come up with one that was not covered in class!)

2 Oblivious Transfer

Our OT protocol only supports 1-out-of-2 OT, which is sufficient for garbled circuits. However, 1-out-of-n OT (for $n > 2$) may also be useful in certain applications. In 1-out-of-n OT, the sender has n messages $m_0, m_1, \ldots, m_{n-1} \in \{0,1\}^\ell$ and the receiver has an input $c \in \{0, 1, \ldots, n-1\}$. From the OT protocol, we would like the receiver to learn only m_c and the sender to learn nothing.

(1) Extend the Diffie-Hellman-based OT protocol to a semi-honest 1-out-of-n OT for an arbitrary n.

Hint: 1-out-of-2 OT should be a special case of the extended protocol.

(2) Why is the above protocol correct?

(3) Why is the above protocol secure against a semi-honest sender?

(4) **(Extra Credit)** Why is the above protocol secure against a semi-honest receiver?

3 Malicious Garbler

As is, our 2PC protocol doesn't protect against malicious adversaries. For instance, a malicious garbler might not garble the circuit that the two parties agreed on.

(1) Consider the private dating example where the two parties want to jointly compute a single AND gate on their private inputs. Describe an attack that allows the garbler to learn the evaluator's input bit (assuming the garbler is malicious and the evaluator is honest).

(2) Suppose we wish to boost security of this protocol by enforcing the garbler to generate each garbled gate honestly; for instance, an AND gate is actually an AND gate and not an OR gate. This way, the evaluator can be sure that the function they're evaluating is the one they agreed on. What cryptographic technique can we use? A single-phrase response suffices.

4 (Extra Credit) GMW for Arithmetic Circuits

In this problem, we extend the GMW protocol to *arithmetic circuits* over the ring $\mathbb{Z}_{2^\ell}$. In particular, each wire w carries a value $v^w \in \mathbb{Z}_{2^\ell}$; the arithmetic circuit consists of ADD and MULT gates for addition and multiplication modulo 2^ℓ.

Throughout the protocol, we keep the invariant that for each wire w, the n parties hold additive secret shares for its value v^w over the ring $\mathbb{Z}_{2^\ell}$, namely, each party holds a random share $v_i^w \in \mathbb{Z}_{2^\ell}$ such that $\sum_{i\in[n]} v_i^w = v^w \mod 2^\ell$.

(1) **Inputs:** For each input wire w, say it carries an input from party P_k with input value $v^w \in \mathbb{Z}_{2^\ell}$, how does P_k generate additive secret shares of v^w and distribute them among all the parties?

(2) **ADD gates:** For each addition gate, the n parties hold additive secret shares $\{a_i\}_{i\in[n]}$ and $\{b_i\}_{i\in[n]}$ for the two input wires with values a and b, respectively. How can they generate additive secret shares $\{c_i\}_{i\in[n]}$ for the output wire with value $c = a + b \mod 2^\ell$?

(3) **MULT gates:** For each multiplication gate, the n parties hold additive secret shares $\{a_i\}_{i\in[n]}$ and $\{b_i\}_{i\in[n]}$ for the two input wires with values a and b, respectively. Now they want to generate additive secret shares $\{c_i\}_{i\in[n]}$ for the output wire with value $c = a \cdot b \mod 2^\ell$. Explain how this problem can be reduced to a **Reshare** protocol (between two parties over $\mathbb{Z}_{2^\ell}$). In the Reshare protocol, two parties hold inputs

$x, y \in \mathbb{Z}_{2^\ell}$ respectively; from the protocol they learn random $r, s \in \mathbb{Z}_{2^\ell}$ respectively such that $r + s = x \cdot y \mod 2^\ell$.

(4) **Reshare:** Design such a Reshare protocol between two parties over $\mathbb{Z}_{2^\ell}$ using 1-out-of-2 OT.

Hint: Consider the bit decomposition of y.