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 6th*

1 Crypto Notions

(1) What does each of zk, S, N, and ARG mean in zk-SNARG? What does K mean in zk-SNARK?

(2) Give a potential application of zk-SNARG/zk-SNARK in practice. (Try to come up with one that was not covered in class!)

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

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

2 Commitment Schemes

Consider a commitment scheme $(\mathsf{Commit}, \mathsf{Open})$ for a finite message space $\mathcal{M}$ (for instance, $\mathcal{M} = \{0,1\}$). Recall the hiding and binding properties:

Hiding: The hiding property states that for any $m_0, m_1 \in \mathcal{M}$, $\mathsf{Commit}(m_0; r)$ is indistinguishable from $\mathsf{Commit}(m_1; s)$ for randomly sampled r and s.

Binding: The binding property states that it is hard for the sender to find $m_0, m_1 \in \mathcal{M}$, $m_0 \neq m_1$ and randomness r, s such that $\mathsf{Commit}(m_0; r) = \mathsf{Commit}(m_1; s)$.

(1) Recall the hash-based commitment scheme. The message space is the set of all strings, namely $\mathcal{M} = \{0,1\}^*$. To commit to a message $m \in \mathcal{M}$, the sender randomly samples $r \leftarrow \{0,1\}^\lambda$ where λ is the security parameter, and computes $\mathsf{Commit}(m; r) = H(r \| m)$. Why is this scheme hiding and binding (under what computational assumptions)?

(2) Recall the Pedersen commitment scheme on a cyclic group $\mathbb{G}$ of prime order q with generator g. Let the receiver first sample a random group element $h \in \mathbb{G}$. The message space is $\mathcal{M} = \mathbb{Z}_q$. To commit to a message $m \in \mathcal{M}$, the sender randomly samples $r \leftarrow \mathbb{Z}_q$ and computes $\mathsf{Commit}(m; r) = g^m \cdot h^r$. Why is this scheme hiding and binding (under what computational assumptions)?

(3) **(Extra Credit)** Recall the Merkle tree construction on a message space $\mathcal{M} = \{0,1\}^{2^d}$. For any message $m \in \mathcal{M}$, we build up a binary tree of depth d where each leaf node is a commitment to a single bit $\mathsf{Com}(m[i])$, where Com is a commitment scheme for bits. Each non-leaf node is computed as $H(\mathsf{lc} \| \mathsf{rc})$ where lc and rc are its left and right children. Finally, we use the root of the tree as the commitment for $\mathsf{Commit}(m)$. Why is this construction hiding and binding (under what computational assumptions)?

3 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?

4 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.