| | |
|---|---|
| **CS65500: Advanced Cryptography** | Instructor: Aarushi Goel |

Homework 4

*Due: March 26; 2025 (11:59 PM)*

# 1 Semi-Honest Secure Multiparty Computation

1. **(20 Points)** Let Alice and Bob be two parties with private inputs $a \in \mathbb{F}$ and $b \in \mathbb{F}$, respectively. They wish to determine whether their inputs are equal, i.e., whether $a = b$, while ensuring that neither party learns any additional information about the other's input.

   **Design a semi-honest secure** two-party computation protocol that outputs 1 to both parties when $a = b$, and 0 otherwise. **Prove that your protocol is secure.** Specifically, formally show that if $a \neq b$, a semi-honest **PPT** Alice should not learn any information about $b$, and a semi-honest **PPT** Bob should not learn any information about $a$.

   (**Hint:** Consider using a semi-honest secure 1-out-of-2 Oblivious Transfer (OT) protocol.)

2. **(20 Points)** Let $P_1, \ldots, P_n$ be $n$ parties, where at most $t < n/2$ are semi-honest. Suppose that a trusted party computes and distributes $(t, n)$ Shamir Secret Shares of a random value $r \xleftarrow{\$} \mathbb{F}$ to these parties. Later, these parties hold $(t, n)$-Shamir secret shares of a value $x \in \mathbb{F}$, meaning each party $P_i$ possesses a share $x_i$, forming a valid $(t, n)$ secret sharing of $x$. They wish to determine whether these are shares of $x = 0$.

   **Design a semi-honest, statistically secure** multiparty computation protocol that outputs 1 to all parties if $x = 0$ and 0 otherwise. **Prove that your protocol is secure.** In particular, formally show that if $x \neq 0$, any **semi-honest unbounded adversary** corrupting at most $t < n/2$ parties does not learn any information about $x$.

   (**Hint:** This is essentially a multiparty analog of the previous problem.)

# 2 Semi-Malicious Security

**(10 Points)** Thus far, we have discussed two types of adversaries: **semi-honest** and **malicious**. Now, consider a new type of adversary, which we call a **semi-malicious adversary**. In this corruption model, the adversary follows the protocol **honestly**, except when choosing randomness, which may be arbitrary.

Consider the **1-out-of-2 semi-honest oblivious transfer protocol** discussed in class. Show an **attack** demonstrating that this protocol is **not secure** against a **semi-malicious receiver**.

# 3 Zero-Knowledge Proofs

**(15 Points)** Assume that $\mathbf{P} \neq \mathbf{NP}$. Show that there exist **non-trivial NP** languages for which **non-interactive proofs** cannot exist if they must satisfy both: **zero-knowledge**, and **efficient verification**. Here, **non-interactive** means that the prover sends a **single-shot proof** to the verifier, without requiring any additional interaction or prior trusted setup.

## 4 Beaver Triples

**(15 Points)** Recall that a **Beaver triple** is a tuple $(a, b, c)$, where $a, b \xleftarrow{\$} \mathbb{F}$, $c = a \cdot b$, and each party receives a secret share of $a, b$, and $c$ (and the actual values $a, b, c$ remain hidden from all parties). As discussed in class, Beaver triples enable **communication-efficient secure multiparty computation (MPC)**. In class, we had assumed that these triples were generated by a "trusted entity" at the start of the protocol.

In this problem, we will explore a secure method to generate Beaver triples without relying on a trusted entity. For simplicity, we focus on the **two-party setting** and generate Beaver triples over the **binary field** $\mathbb{Z}_2$. Throughout this problem, assume that Alice and Bob are **semi-honest**.

**Show how Alice and Bob can securely generate a Beaver triple using Yao's garbled-circuit-based two-party protocol**. Your construction should not modify the internal details of Yao's protocol (in fact, any secure two-party computation protocol could be used here). Then, provide an **informal argument** explaining why your protocol is both **correct** and **secure**.

(**Hint:** To apply Yao's protocol, define a two-party functionality $f$ that Alice and Bob will compute jointly. Consider letting Alice's inputs to $f$ be her shares $(a_1, b_1, c_1)$ of the Beaver triple, which she samples uniformly at random at the beginning of the protocol.)