Secure Auction System Implementation

Introduction

The task required implementing a secure auction system using Obliv-C that matches 3 buyers with 3 sellers while preserving privacy. Each buyer has a maximum price they're willing to pay, and each seller has a minimum price they're willing to accept. The system must find appropriate matches and determine fair pricing without revealing sensitive information.

Allocation Method

I implemented a greedy allocation algorithm that processes buyers in order (B1, B2, B3). For each buyer, the algorithm attempts to match with available sellers in order (S1, S2, S3), subject to two key constraints:

1. A seller can only be matched with one buyer
2. A buyer can only be matched with a seller if the buyer's maximum price (Yi) is greater than or equal to the seller's minimum price ($X[R_i]$)

This approach, while simple, effectively finds valid matches while respecting all constraints. The greedy priority gives precedence to earlier buyers and sellers, which creates a deterministic matching pattern. I chose this method because:

1. It provides a straightforward implementation in Obliv-C.
2. It guarantees that no buyer is left unmatched if a viable seller exists.
3. It avoids the complexity of more sophisticated algorithms while still producing valid allocations.

Pricing Method

For the pricing strategy, I implemented a fair split between the buyer's maximum and the seller's minimum:

$$C_i = (Y_i + X[R_i]) / 2$$

This approach offers several advantages:

1. It divides the "surplus" equally between buyer and seller.
2. Neither party has an incentive to misrepresent their values.
3. The calculation is straightforward to implement securely.

Security Considerations

All computation occurs within Obliv-C's secure environment, ensuring that private values remain protected throughout the matching process. Only the final results (R1, R2, R3, C1, C2, C3) are revealed at the end of the protocol.

The implementation takes care to:

- Maintain all values as obliv variables during computation.
- Only reveal final results.
- Use secure conditional operations for matching logic.

Information Leakage Analysis

An important aspect of secure multiparty computation is understanding what information is leaked to participants. Here's my analysis of what each party learns under different scenarios:

- **When shares of R_i are sent to B_i :**
  Each buyer Bi learns only their own match result (which seller they're matched with, if any). They don't learn other buyers' matches or any seller's minimum price. This protects both the privacy of other buyers and the sellers' pricing strategies.

- **When i such that R[i]=j is sent to Sj:**
  Each seller Sj learns only which buyer they're matched with (if any). They don't learn other matches or any buyer's maximum price. This protects both the privacy of other sellers and the buyers' valuation of the items.

- **When shares of Yi are sent to Bi:**
  This reveals no new information since each buyer already knows their own maximum price.

- **When shares of Yi such that Ri=j are sent to Sj:**
  This would reveal the matched buyer's maximum price to the seller, which is a significant privacy leak. The seller would learn exactly how much the buyer was willing to pay, which could be exploited in future negotiations. My implementation avoids this by not sharing this information directly.