Secure Vickrey Auctions with Rational Parties

Chaya Ganesh, Shreyas Gupta, Bhavana Kanukurthi, and Girisha Shankar

Indian Institute of Science
{chaya,shreyas17521,bhavana,girishabs}@iisc.ac.in

Abstract. In this work, we construct a second price (Vickrey) auction protocol (SPA), which does not require any auctioneers and ensures total privacy in the presence of rational parties participating in the auction. In particular, the confidentiality of the highest bid and the identity of the second highest bidder are protected. We model the bidders participating in the second price auction as rational, computationally bounded and privacy-sensitive parties. These are self-interested agents who care about winning the auction more than learning about the private bids of other parties. A rational party does not deviate from the protocol arbitrarily but does so only for its own individual 'advantage' – without any consideration for others. Such an advantage is modelled using suitable utility functions.

We show that for rational and computationally bounded parties participating in our second-price auctions protocol, there exists a privacy-preserving dominant strategy equilibrium in which every party prefers to follow the protocol rather than to deviate.

Our protocol is implemented using open-source cryptographic constructs. Running our SPA protocol on commodity hardware with 15 bidders, with bids of length 10 bits, completes in 1.26sec and has total communication of 0.77MB whereas, under similar conditions, Atlas (semi-honest) protocol takes 40% more time (2.11 sec) and 87% more communication (6.09MB).

Table of Contents

1 Introduction

Auctions are mechanisms where buyers compete to purchase goods by bidding. Auctions are characterized by their ability to efficiently allocate goods to those buyers who value them the most while fetching profit to the sellers. Digital auctions are a billion-dollar industry with companies like Google, Meta, etc. running as many auctions daily in order to auction out their ad space. Almost every sector makes use of them – auctions are used for fundraising, spectrum allocations by governments, and trading commodities in financial markets. Typically, it is the highest bidder who wins the auction. Depending on the price paid by the winner of the auction, we can have either *first price auctions* where the winner pays the highest bid or *second price auctions* (Vickrey auctions) where the winner pays the second highest bid.
Second price auctions (SPA) are known to be strategy-proof; that is, while participating in second price auctions, the bidders are incentivized to bid their true valuation [Kri09], without considering the bids of other parties. This is in contrast to first price auctions, where bidders may choose their bids strategically.

Second price auctions satisfy the requirements of both auctioneer and bidders. The auctioneer is able to elicit true valuations from bidders and, hence, sells the item to the bidder who values the item most. The winning bidder gets to purchase the auctioned item at a discount (compared to its valuation). Often, an important goal for auctions is to ensure the privacy of bids: for this, there are sealed bid auctions. Privacy is especially critical for second price auctions, where bidders use their true valuation as their bids. The bidders may not want to divulge any information about their bids. In particular, the highest bidder, too would not want to divulge its own bid. Similarly, while the highest bidder pays the bid of the second highest bidder, the latter would wish to keep its own identity secret. A secure second price auction protocol should only output the winner's identity and the second highest bid. The protocol should protect the confidentiality of the highest bid, the identity of the second highest bidder, and losing bids.

When auctions are executed by trusted auctioneers, it is trivial to ensure such privacy requirements. However, such trust is difficult to realize in practice. Moreover, even when privacy is ensured, second price auctions are vulnerable to manipulation by a corrupt auctioneer who can enhance its utility by declaring a higher value for the second highest bid. Hence, designing a protocol that can function without needing an auctioneer is desirable.

All existing protocols for second-price auctions either need a trusted auctioneer or use generic Multi-Party Computation (MPC) protocols; therefore, they assume some parties to be completely honest. *In this work, we build a second price auction protocol secure in the rational setting that guarantees privacy and does not require any trusted auctioneer.*

Rational security model. In traditional cryptography, every party is either honest or malicious. However, there are many scenarios where parties are simply rational agents who have a clearly specified end goal they are trying to achieve. In the case of auctions, for example, it is fair to assume that the primary goal of parties (bidders) is to win the auction. At a high level, a rational party deviates from the protocol only if it has something to 'gain'. This gain is defined by a utility function associated with the party. Assuming all parties are rational, any party participating in an auction protocol would do so to win it. If they cannot win the auction, a secondary goal of a party may be to learn about the bids of other parties involved in a sealed bid auction.

Our contribution. This work presents a concretely efficient protocol for *Second Price Auction* with guaranteed privacy. Our protocol achieves *Computational Weakly Dominant Strategy Equilibrium* for rational, computationally bounded parties without the need for any trusted auctioneer to maintain the privacy of bids. In our security model, the bidders are modelled as privacy-sensitive, self-interested agents using appropriate utility functions to capture their motivation to maximize their individual gains. Our protocol ensures that a) being honest is a weakly dominant strategy equilibrium for parties b) when parties stay honest, the protocol leaks nothing except for the output and c) parties have no incentive to form collusion to either gain monetarily or learn private inputs of other parties.

We stress that the rational security model is *incomparable* with standard secure computation models like semi-honest and malicious. Notably, the rational model does not insist on honest behavior from every party. However, in both semi-honest and malicious models, there is a single adversary controlling a certain threshold of parties. Such a threshold assumption implies that there should exist at least some honest parties in the protocol. In contrast, the rational security model is more realistic, and there need be *no honest parties* in it; all parties are rational and may act strategically for better gains.

We have implemented our SPA protocol in C++, using OpenSSL and Boost open-source libraries. Running our rational secure protocol on a commodity hardware, with 15 bidders, 10-bit length bids resulted in 0.77MB communication and took 1.26s. We show that our protocol is concretely more efficient than secure second price auction implementations realized using generic MPC protocols.

Outline. The rest of the paper is organized as follows. Related works are presented in Section 1.1, where we provide a survey of prior works related to rational security and auctions. Section 1.2 presents an overview of our work, specifically highlighting the challenges and design decisions for the SPA protocol. Preliminaries are described in Section 2. Section 3 describes the rational security model used in our work. Section 4 provides a detailed description of the protocol divided into multiple phases. We analyze the strategy space available to rational parties along with security proofs in Section 6. For each strategy, we argue why it is rational for parties to stay honest. In Section 6.5 we introduce our modelling of collusion and analyze the same in greater detail. Lastly, we summarize the experimental results in Section 7 and conclude.

1.1 Related work

Game theoretic tools for cryptography. The intersection of game theory and cryptography has attracted interest from the cryptographic research community for the past two decades. Dodis, Halevi and Rabin [DHR00] initiated a line of work capturing notions of incentives in cryptographic definitions. Halpern and Teague [HT04] considered secret sharing and secure computation in the rational setting. They define a solution concept that is a variant of Nash equilibrium. Subsequent works define further variants of equilibrium, like computational versions, for modeling collusion and mechanism design [IML05,ADGH06,GK06,Hal08,KN08,HP10,Mic14,DM16]. A noteworthy contribution in rational cryptography is the Rational Protocol Design (RPD) framework introduced by Garay, Katz, Maurer, Tackmann and Zikas [GKM+13] and employed subsequently in [GKTZ15,BGM+18]. RPD models the protocol design as a two party game between a protocol designer and an external attacker, where the attacker's goal is to break security properties, and the goal of the protocol designer is to prevent the attacker from succeeding. Biçer, Yildiz and Küpçü [BYK21] make use of *Weakly dominant strategy* for coalitions in their work to develop the notion of *m-stability* which offers threshold security against a coalition of size m.

Auctions. Miltersen, Nielsen and Triandopoulos [MNT09] defined a rational security framework for first price auctions wherein the bidders care more about the monetary payoffs than learning about bids of other parties. Important contributions of their work are a novel notion of *information utility* and *Privacy enhanced computational Nash Equilibrium* for modelling rational security. They use generic MPC protocols to run the first price auction and show that their protocol achieves this equilibrium. However, Nash equilibrium ensures honesty for a rational party only when all other parties are also honest. *Instead, we make use of a stronger notion of dominant strategy equilibrium wherein honest strategy is the best response irrespective of others' strategic choices.* Moreover, our solution concept also guarantees privacy by exhibiting a simulator.
In SEAL [BHSR19], Bag, Hao, Shahandashti and Ray propose auction protocols without needing auctioneers. They make use of *Anonymous Veto Protocol* (AVP) [HZ06]. FAST [DGP22] is a similar protocol by David, Gentile and Pourpouneh, which also uses AVP. However, both these works have a non-trivial leakage. Their first price auction protocol leaks the first few bits of the second highest bid, and their second price auction protocol leaks the first few bits of the highest bid. Ganesh, Kanukurthi and Shankar [GKS22] have introduced a first price auction protocol without requiring auctioneers that also uses a variant of AVP, but ensuring total privacy in a rational setting.

Second price auctions. Brandt introduced the protocol for second price auctions in [Bra01]. In this protocol, the auctioneer is privy to bids from all parties. This is avoided in the work of Kurosawa and Ogata [KO02], who use bit-slice approach with multiple auctioneers, the majority of whom are honest, to determine the highest bid, bit by bit. Nojoumian and Stinson [NS14], use a setting with m bidders and n auctioneers in addition to a trusted party.
Naor, Pinkas and Sumner [NPS99] require the role of an auctioneer for running the auction, apart from the auction-issuer who sets up the auction. These two entities are expected not to collude. Their protocol uses garbled circuits prepared by the auction-issuer. In a similar setting, Lipmaa, Asokan and Neimi [LAN02] introduce a second price auction protocol which requires the roles of auctioneer and seller who do not collude. They use a pre-specified set of bids to run the auction. In Boaz and Herzberg [CH13], the auctioneer is not trusted. Instead, a trusted supervisor ensures that the auctioneer does not deviate by running random checks on the computation. Rational setting is used to argue the security.
Omote et al. [OM03] introduced a protocol using two non-colluding auctioneers – AM_1 and AM_2. AM_1

is used during the setup stage and AM_2 for the computation stage. Similar to Lipmaa et al. [LAN02], bidders can use only pre-specified points as bids. Another similar work is by Hso and Miyaji [HM21], who use smart contracts to run generalized Vickrey auctions on a pre-specified number of bid points.
All works mentioned above require the presence of one or more auctioneers and/or some additional entities for running the auction. In some cases, bidders are allowed to choose the bids only from a pre-specified set of bids. Our work stands out in that it does not require any auctioneer and has flexibility for bidders to choose their bids while offering full privacy. To the best of our knowledge, ours is the first such protocol to implement second price auctions in a rational security model.

1.2 Technical overview

Threat model. In our rational security model, every party may deviate from the protocol to meet their most desired goal of winning the auction. Other considerations, such as learning information about other parties, do exist but are secondary. Parties can either act in their individual capacity or collude strategically to enhance their gains. The utility function captures these considerations.

FPA to SPA: Challenges. In first price auctions (FPA), the auction is won by the highest bidder who pays the highest bid. Therefore, it suffices to design a secure computation protocol for the *"max"* function. A trivial, though insecure, solution for SPA would be to run the FPA till the end to identify the winner and then rerun the auction without the winner's participation to identify the second highest bid. (In bit-by-bit protocols for *max*, it would suffice to run the protocol until the highest bidder is identified and then rerun the computation appropriately. In fact, this is precisely the approach used in SEAL [BHSR19] and FAST [DGP22].)

Such approaches using FPA to compute SPA are insecure as they would reveal non-trivial information about the highest bid. Recall that for SPA, we need to keep private a) the value of the highest bid and b) the identity of the second highest bidder. Therefore, the main challenges in building SPA are two-fold: *How do we identify the highest bidder without revealing their bid? How do we compute the second highest bid without revealing bidder's identity?*

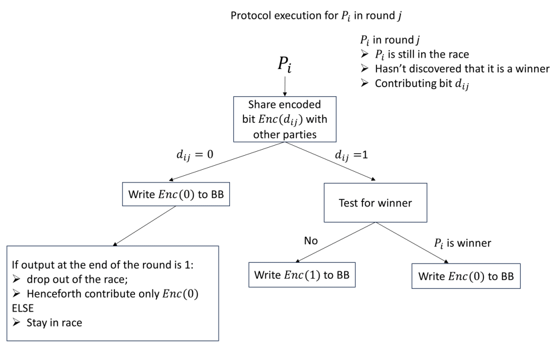
We overcome these challenges by designing an "oblivious winner discovery" (OWD) mechanism. In OWD, the parties communicate with each other so as to learn only if they are the highest bidders. The fact that the highest bidder has made this discovery remains oblivious to all other parties till the end of the protocol. Once this discovery happens, the highest bidder effectively drops out from the *max* computation. Therefore, the second highest bidder believes that it has still not lost and thus, the protocol reveals the value of the second highest bidder. Here's the tricky part: once the highest bidder drops out, the second highest bidder may, at some point, wrongly interpret that it is the highest bidder as every party runs their own winner discovery. We need to come up with a mechanism where the highest bidder not only drops out of the race but then actively needs to trick the second highest bidder into believing that there may be a higher bidder who continues to be in the race.

We accomplish this by having a two step process. Each party first checks to see if they have the highest bid via an interactive process. Once the highest bidder discovers that it has the highest bid, it "tricks" in this interaction with the other bidders: whenever they have a '1' to contribute, it convinces them that it too has a '1', effectively sabotaging their winner discovery phase. We now offer more details.

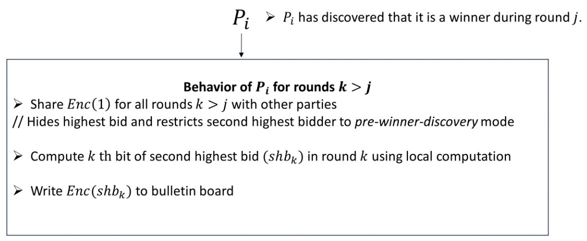
Anonymous Bidding Protocol (ABP). This is a sub-protocol that computes the *max* function bit-by-bit using a bulletin board, as described next. The parties start by contributing bits from their bids, MSB onward. In the first round, they learn the logical OR of the first bit of parties' individual bids. If a party has contributed a 0, but the output of the first round was 1, it does not have the highest bid and hence drops out of the "race". However, such a party continues to participate[1] in the protocol and contributes a 0 bit for all subsequent rounds of computation. This process continues, with parties dropping out of the race once they realize they no longer have the highest bid. Thus, the computed output is the *max* of all bids. This sub-protocol is called the Anonymous Bidding Protocol (ABP) (see Appendix A). In our work, we use ABP, which has also found applications in prior works such as SEAL [BHSR19], FAST [DGP22] and [GKS22].

For the case of SPA, our goal is to compute the second highest bid and not the *max* value – i.e., we cannot use ABP as is. To enable computation of the second highest bid, the highest bidder, too, needs to drop out of the race during the ABP run. However, this poses few challenges: *a) we need to ensure the highest bidder's dropping off must be concealed from all other parties (OWD). b) once the highest bidder drops off, the second highest bidder should not conclude incorrectly that it is the highest bidder.*

[1] The security-deposit collected up front in the protocol ensures such participation


Protocol execution for P_i in round j
P_i in round j
➢ P_i is still in the race
➢ Hasn't discovered that it is a winner
➢ Contributing bit d_{ij}
P_i
Share encoded bit $Enc(d_{ij})$ with other parties
$d_{ij} = 0$
$d_{ij} = 1$
Write $Enc(0)$ to BB
Test for winner
If output at the end of the round is 1:
➢ drop out of the race;
➢ Henceforth contribute only $Enc(0)$
ELSE
➢ Stay in race
No
P_i is winner
Write $Enc(1)$ to BB
Write $Enc(0)$ to BB

(a) Protocol flow for an arbitrary party


P_i
➢ P_i has discovered that it is a winner during round j.
Behavior of P_i for rounds $k > j$
➢ Share $Enc(1)$ for all rounds $k > j$ with other parties
// Hides highest bid and restricts second highest bidder to pre-winner-discovery mode
➢ Compute k th bit of second highest bid (shb_k) in round k using local computation
➢ Write $Enc(shb_k)$ to bulletin board

(b) Protocol flow post winner-discovery

Fig. 1

We overcome the first challenge by leveraging one key property of the ABP protocol: a party that contributes a 1 bit can compute logical OR of contributing bits of other parties. In fact, this is the cause of leakage mentioned above in SEAL [BHSR19] and FAST [DGP22]. However, we utilize this leakage to build a "winner discovery mechanism" where the highest bidder can (privately) learn that it has the highest bid. We enable *only* the winner to obtain the leakage information while other parties obtain dummy values (which are required to prove honest computation in the end). For this, we make use of Oblivious Transfer (OT), with the contributing bit used as OT choice bit. To overcome the second challenge, the winner needs to keep the second highest bidder in the race. For this, the winner contributes bits to the protocol in such a way that the second highest bidder neither drops out of the race nor assumes it is the winner. To ensure that the second highest bidder does not drop out of the race, the winner contributes the same bit as that of the second highest bidder and writes it to BB. This requires the winner to learn the bit being contributed by the second highest bidder. The winner accomplishes the same by using the OT choice bit to be 1 for all rounds after the winner discovery. In order to ensure that the second highest bidder does not consider itself to be the highest bidder, the winner sends encoding of 1 bit to all parties after the winner discovery. This thwarts the second highest bidder from assuming that it is the winner.

Here is a high-level overview of our protocol. We begin by listing its salient features :

1. The protocol runs for l rounds, starting from the MSB of the bid. Here l is the number of bits used in binary representation of bids.
2. There exists a unique $\bar{j} \in [l]$ (winner discovery round) such that bits of the second highest bid are the same as bits of the highest bid for the first $\bar{j} - 1$ rounds. In $\bar{j}$th round, the highest bid has 1, and the second highest bid has 0.
3. In all the rounds after $\bar{j}$th round, the highest bid identified in $\bar{j}$th round is ignored for further computation.

Now we present the overview of the SPA protocol, which is divided into 3 phases. In the following we consider n parties, $(P_1, \ldots, P_n)$ with respective bids $(b_1, \ldots, b_n)$. jth bit of b_i is denoted by b_{ij} and P_i's contribution during round j is denoted as d_{ij}.

Set-up Phase: Each party P_i registers for auction by paying a deposit D and receives the public parameters pp. P_i commits to bits of its bid – used to generate NIZK proofs to verify the correctness of computations of the losing bidder.
Auction Phase: The protocol proceeds for l rounds. A typical round j for an arbitrary party P_i is depicted in Figure 1a. In each round, parties run the SPA protocol as follows.

– Exchange of encoded bits: Every pair of parties communicates privately with each other using OT. To receive an encoded bit from P_k in round j, as an OT receiver, P_i uses d_{ij} as its choice bit. As a sender, P_k uses encoding of d_{kj} as $M^{(1)}$ and a 0-token as $M^{(0)}$ for OT.
– Winner discovery: Each party checks if it is the only party contributing 1 in that round. As the OT receiver, every party gets the encoded bits from other parties and deduces that it has won if

PRODUCT of other parties' encoded bits is 1 (see Appendix A). If so, party transitions to WINNER state.

- POST WINNING ACTIONS: (Refer to Figure 1b.) Once P_i knows it has won the auction during round $\bar{j}$, it uses $d_{ij} = 1$ in all subsequent OT rounds $j = \bar{j}$ to l, both as choice bit and for generating $M^{(1)}$. This design choice helps the winner to obtain the contribution bit of the second highest bidder during each round j. This also prevents the second highest bidder from assuming that it has the highest bid.
- Every party writes the encoding of its contribution bit to the BB so that all parties involved in the protocol can compute the output of the current round. The winner (if already discovered) writes the same bit as that of the second highest bidder.

VERIFICATION PHASE:

- The winner comes forward to claim the auction by producing the private randomness used for encoding the bits.
- Each losing party provides NIZK proof of correct computation for each round. It also produces the 0-tokens for the rounds which have computed output to be 0 to prove the correct usage of OT choice bit.
- In case there are multiple claimants for the auction – which can happen if there are deviating parties, the protocol has a winner resolution mechanism. In this case, each claimant needs to open all OT message randomness, thus exposing the cheating party.
- Whenever a party's deviation is detected, the protocol terminates, and that party forgoes its deposit, which gets distributed among honest parties.

Security. In order to argue security, we note that a rational party sees value in a) increasing its monetary utility by winning the auction and b) learning information about other players' inputs. To capture these, we use monetary and information utilities (similar to earlier works in [MNT09,GKS22]). We also use the notion of *Privacy preserving computational dominant strategy equilibrium* for analyzing the privacy concerns of rational parties. We show that as long as parties value monetary utility much more than information utility, they have no incentive to deviate from the protocol. Then, for parties that do not deviate from the protocol, we argue privacy using the simulation paradigm.

Handling collusion. Rational parties can collude strategically to enhance their utilities. We consider privacy-sensitive parties who value their individual privacy over learning the private inputs of other parties. Such parties prefer to learn about the private inputs of other parties only if there is no loss of individual privacy in the process. Our modelling has two important consequences: a) parties who are not part of collusion are not restricted, to be honest – but rational, strategically deciding to stay out of collusion. b) this model helps to get collusion-resistance for free in our protocol.
In our setting, a group of privacy-sensitive parties come together to form a collusion if each of them is better off as a part of the group rather than acting alone. We believe that such a notion of collusion is more natural in a rational framework. Moreover, we consider the formation of collusion to take place *only after* the bid values are chosen (and committed). In particular, collusion formed even before the first message of protocol gets sent is beyond the scope of the protocol. However, we would like to emphasise that collusion can be formed at any time during the protocol or after the protocol ends. Using our model, we argue that no collusion is rational as opposed to providing a threshold guarantee.

2 Preliminaries

Notation. We denote the security parameter by λ. Let $\mathbb{G}$ be the description of the group of prime order q, and with generators g, h. A function negl is said to be negligible if $\mathsf{negl}(n) < 1/p(n)$ for all positive polynomial functions $p(\cdot)$ and for all $n > n_0$ for some $n_0 \in \mathbb{N}$. We denote *Probabilistic Polynomial Time* by PPT. We also use $\approx_c$ to denote computational indistinguishability between two probability distributions.

2.1 Building blocks

Here we describe some key building blocks that are used in our protocol.

Definition 1 (Commitment scheme). *Let* $\mathsf{M}, \mathsf{C}, \mathsf{R}$ *denote the message space, commitment space and randomness space respectively.*
A commitment scheme consists of a tuple ($\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}$*) of PPT algorithms where:*

– $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *generates public parameters* pp.
– $\mathsf{Com}(m, r) \to c$ *takes input* pp *(implicitly), message* $m \in \mathsf{M}$*, randomness* $r \in \mathsf{R}$ *and outputs a commitment* $c \in \mathsf{C}$.
– $\mathsf{Open}(c, m, r) \to B \in \{0, 1\}$ *checks if the commitment* c *opens to the message* m*, given randomness* r*. If so, outputs* 1 *and* 0 *otherwise.*

The security of a commitment scheme guarantees two properties: *hiding* and *binding*. Informally, the hiding property guarantees that for any two messages m_0 and m_1, no PPT algorithm can distinguish between commitments to m_0 and m_1. The binding property guarantees that no PPT algorithm can open a commitment to two different messages.

Oblivious transfer. Oblivious Transfer (OT) is a two-party protocol with sender S having two secret messages and receiver R with a single choice bit. The goal of the protocol is for R to learn the message of its choice without learning about the other message from the sender. In addition, OT protocol demands that S does not learn about the choice exercised by R. Let M, R be the message space and randomness space, respectively. An OT protocol proceeds as follows for two PPT parties, R and S:

– $\mathsf{OT.R}_1(\alpha, \beta) \to (\mathsf{otr}^1, \mathit{state})$ is invoked by R with inputs: choice bit $\alpha \in \{0, 1\}$, randomness $\beta \in \mathsf{R}$. otr^1 is the first message sent by R to S. state is the internal state of R.
– $\mathsf{OT.S}(\mathsf{otr}^1, M^{(0)}, M^{(1)}, \gamma) \to \mathsf{ots}_{S,R}$ takes otr^1, messages $M^{(0)}$, $M^{(1)} \in \mathsf{M}$, randomness $\gamma \in \mathsf{R}$ and outputs message $\mathsf{ots}_{S,R}$ to be sent by S to R.
– $\mathsf{OT.R}_2(\mathsf{ots}_{S,R}, \mathit{state}) \to M^{(\alpha)}$ Invoked by R with $\mathsf{ots}_{S,R}$ from S and internal state to retrieve the message $M^{(\alpha)}$.

The security of OT protocol ensures that the receiver does not learn about $M^{(1-\alpha)}$ and the sender does not learn about α.

Bulletin board. The *Bulletin Board* (BB) is an abstraction for an authenticated broadcast channel with memory. We assume the existence of such a broadcast channel for our protocol. We do acknowledge that the implementation of such broadcast channels can have different threat models than ours. However, in this work we do not concern ourselves with the problem of handling broadcast in a rational setting, which is of independent interest and can be considered for future works. In our protocol, parties can write messages on to the BB for public consumption. The BB is expected to satisfy the following properties: (i) Every message written on the BB is associated with a unique party and is readable by all other parties. (ii) The messages written on BB are immutable.

A BB can be realized through a public chat-room, shared web-page, or even a private blockchain. Our protocol does not rely upon any specific realization of BB.

Contract Functionality. Our protocol uses security deposits from the bidders to enforce honest behavior. Any deviation from a bidder that affects the protocol described in Section 4.2 results in the forfeiture of the security deposit, which is redistributed among the non-deviating parties. Such a penalty mechanism requires the use of contracts with the parties in the auction and assumes the use of certain *transferable utility* (TU) among parties. Such contracts can be drawn within a legal framework or they could be smart-contracts on a block-chain. If the contracts are using the traditional legal framework, then the TU can be realized using fiat currency that is common among all parties in the protocol. In such a case, there needs to be a trusted entity to receive, manage and redistribute the deposits. Such a trust can also be a threshold trust wherein a set of entities need to come together to operate on the deposits. On the other hand, if one were to make use of smart-contracts on a block-chain, then the native currency (coin) of the block-chain can serve as the TU among the parties. In this case, the smart-contract would receive the deposits from parties before the protocol. At the end of the protocol, depending on the proofs submitted, would either return the deposits for an honest run of protocol or redistribute the deviating parties' deposits when cheating is detected. Schwartzbach [Sch22] proposes a mechanism for realizing payments to facilitate deposits in a rational setting.

2.2 Equilibrium notions

We now describe some game theoretic notations and definitions used in our work. We assume that there are n parties $(P_1, \ldots, P_n)$ participating in a game.

Definition 2 (Normal Form Game [Kat08]). *A normal form game is a tuple* $\{\{\Gamma_i\}_{i=1}^n, \{U_i\}_{i=1}^n\}$ *where for each party* P_i*, a space of possible actions* Γ_i *along with a utility function* U_i *are specified.*

Each party P_i can be associated with a certain strategy $\pi_i \in \Phi$ where Φ is a strategy space available for the parties to choose their strategies from while playing the game. π_i is essentially an algorithm that takes as input the private inputs of P_i, the current state of the game and outputs the action $s_i \in \Gamma_i$ to be taken by P_i. We denote the outcome of the game using the strategy profile of parties (π_i, π_{-i}) where $\pi_{-i} = (\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n)$. We also assign utility functions $U_i(\pi_i, \pi_{-i})$ to each party P_i. These functions represent the perceived utility of different outcomes of the game for the party. We say that a party P_i *prefers* a certain outcome (π_i, π_{-i}) over another outcome (π'_i, π_{-i}) if and only if $U_i(\pi'_i, \pi_{-i}) < U_i(\pi_i, \pi_{-i})$.

Definition 3 (Dominant Strategy [Kat08]). *Given a normal form game: $\{\{\Gamma_i\}_{i=1}^n, \{U_i\}_{i=1}^n\}$ we say, $\pi_i \in \Phi$ is a* Dominant Strategy *for P_i if $U_i(\pi'_i, \pi_{-i}) < U_i(\pi_i, \pi_{-i})$, $\forall \pi'_i (\neq \pi_i) \in \Phi$ and $\forall \pi_{-i} \in \Phi^{n-1}$.*

Such a strategy π_i guarantees that a party P_i can accrue the best utility among all strategies available to it. In the above case, π'_i is also termed as *Dominated Strategy*. The parties typically avoid dominated strategies, whereas dominant strategies are pursued. We also have a weaker notion of *Dominant Strategy* known as *Weakly Dominant Strategy*.

Definition 4 (Weakly Dominant Strategy [Kat08]). *Given a normal form game: $\{\{\Gamma_i\}_{i=1}^n, \{U_i\}_{i=1}^n\}$ we say $\pi_i \in \Phi$ is a* Weakly Dominant Strategy *for P_i if $U_i(\pi'_i, \pi_{-i}) \leq U_i(\pi_i, \pi_{-i}), \forall \pi'_i (\neq \pi_i) \in \Phi, \forall \pi_{-i} \in \Phi^{n-1}$. In addition, $\forall \pi'_i (\neq \pi_i)$ there exists some $\pi_{-i} \in \Phi^{n-1}$ such that $U_i(\pi'_i, \pi_{-i}) < U_i(\pi_i, \pi_{-i})$.*

Definition 5 (Weakly Dominant Strategy Equilibrium (W-DSE) [Nar14]). *For a normal form game $\{\{\Gamma_i\}_{i=1}^n, \{U_i\}_{i=1}^n\}$, the strategy profile $\pi = (\pi_1, \ldots, \pi_n) \in \Phi^n$ is a* Weakly Dominant Strategy Equilibrium *if $\forall P_i, i \in [n]$, π_i is a* Weakly Dominant Strategy *for party P_i.*

A *Dominant Strategy Equilibrium*, whenever it exists, guarantees that every party has a unique *Dominant Strategy* available to it. Thus, each party can realize maximum utility by adopting its *Dominant Strategy*. Since such a strategy becomes a preferred choice for every party, irrespective of the strategic choices of other parties in the game, the chosen strategy profile is an equilibrium.

3 Rational Security model

We model the second price auction to be a game in which participating bidders are rational PPT parties with their individual utility functions. A rational party does not deviate from the protocol arbitrarily but does so only for its own individual 'advantage' – without any consideration for others.

We consider a setting in which parties value their individual privacy over learning the private inputs of other parties. This means that parties would prefer to learn about the private inputs of other parties only if there is no loss of individual privacy in the process.

Moreover, for the case of auctions, our utility function will capture the view that the parties are self-interested agents who care most about winning the auction. A *secondary* incentive may be to learn about the private bids of other parties. To formalize this, we consider monetary utility and information utility [MNT09]. We assume that parties pay a deposit, which they may lose if they are detected to be deviating. The monetary utility is calculated based on this cost and the monetary gain from winning the auction. The information utility captures the fact that parties prefer to reveal as little as possible about their inputs while learning as much as possible about other parties' inputs.

Since we are considering rational parties, we let the parties choose suitable strategies for enhancing their utilities. The protocol demands that parties choose the honest strategy and follow the protocol. However, depending on their utilities, rational parties may choose deviating strategies as well. Moreover, parties can strategically collude to either make monetary gains or to learn about the private bids of other parties.

We are interested in those strategies which are self-enforcing on the parties. Such a state, where parties do not have any incentive to deviate from their chosen set of strategies, is also referred to as an *equilibrium*. To demonstrate security in the rational setting, we first show that following the protocol is a weakly dominant strategy equilibrium. Consequently, to show privacy, it suffices to show that when parties do follow the protocol, they learn nothing beyond the output of the protocol. We show this using the existence of a simulator in the ideal world-real world paradigm.

Privacy-preserving (Computational) Dominant Strategy Equilibrium. Our definition of rational security is identical to the one introduced in the context of first price auctions by Ganesh et al. [GKS22]. Specifically, they define the solution concept called *Privacy enhanced computational dominant strategy* *equilibrium* (PECDSE) for analyzing rational security. In short, parties participating in a game using a cryptographic protocol prefer to follow the protocol honestly while learning nothing more about the private inputs of other parties than what the protocol outputs. Notice that this is not just about the equilibrium but encompasses the security property of the protocol.

While we make use of this solution concept to show the rational security of our protocol, we rename the definition to *Privacy preserving computational dominant strategy equilibrium* (PPCDSE) for reasons that we now explain. The PECDSE definition was motivated by the definition of *Privacy enhanced Nash equilibrium* defined in [MNT09]. However, compared to the definition in [MNT09], PECDSE definition features an added condition for privacy (point (2) in the definition). Consequently, contrary to what one would expect, replacing Nash equilibrium in the definition from [MNT09] with dominant strategy equilibrium *does not* lead to PECDSE.

To resolve this ambiguity, we rename the PECDSE solution concept to Privacy preserving computational dominant strategy equilibrium (PPCDSE).

Definition 6 (Privacy preserving computational dominant strategy equilibrium). *Let $(P_1, \ldots, P_n)$ be a set of rational PPT parties with their respective efficiently computable utility functions U_i, while participating in a n-party protocol Π which computes the functionality $\mathcal{F}$. Let Φ denote the space of strategies. Let $\pi_i \in \Phi$ be the strategy for P_i of following Π. Let $\pi'_i \in \Phi$ be an arbitrary, efficiently computable strategy.*

We say that Π is a Privacy preserving computational dominant strategy equilibrium *if the following hold.*

1. *Π is a* Weakly Dominant Strategy Equilibrium *(W-DSE) with probability* $(1 - \mathsf{negl}(\lambda))$, *where λ is the security parameter; i.e.,*
$$U_i(\pi'_i, \pi_{-i}) \leq U_i(\pi_i, \pi_{-i})$$
for all arbitrary efficiently computable $\pi_{-i} \in \Phi^{n-1}$.
2. *When every $P_i, i \in [n]$ uses $\pi_i \in \Phi$ as the strategy following Π, for each P_i there exists a simulator $\mathcal{S}_i^{\mathcal{F}}$ such that the view of P_i in a real execution of the protocol is computationally indistinguishable from the output of the simulator:*
$$\mathsf{View}_i^{\Pi} \approx_c \mathcal{S}_i^{\mathcal{F}}$$

where View_i^{Π} is the random variable of the transcript of P_i in the protocol Π. The probability is over the choice of private random coins of the parties.

Thus, in order to show rational security, we show that the following properties hold:

1. Equilibrium (Part (1) of Definition 6). We first show that the dominant strategy for parties is to follow the protocol. This property relies on the security of the bit encoding scheme, binding property of the commitment scheme, malicious security of the OT protocol Π_{OT} and soundness of NIZK proofs.
2. Simulation (Part (2) of Definition 6). When parties follow the protocol, we show privacy using the simulation paradigm. Specifically, we show the existence of a simulator for every party such that the view of the party in the real world is indistinguishable from the output of the simulator. Here, we use the hiding property of the commitment scheme, semi-honest security of the OT protocol, security of the bit encoding scheme (14) and zero-knowledge property of NIZK proofs.

3.1 Utilities

We assume there are n parties that participate in the auction protocol modelled as the game, denoted by $P_1, P_2, \ldots, P_n$. Each party P_i has its bid denoted by b_i, with a corresponding l-bit decomposition $(b_{i1}||b_{i2}||\ldots||b_{il})$. Let

- v_i: Perceived private valuation of the auction item for P_i.
- b_i: bid of party P_i. Each bid is a l bit integer. Note that, for second price auctions, rational bidders prefer to use $b_i = v_i$ [Kri09]. We denote $\mathbf{b} = (b_1, \ldots, b_n)$.
- w denotes the index of the winning party, and s denotes the same for the second highest bidder.
- $U_i : \Phi^n \mapsto \mathbb{R}$ is the monetary utility function of a party P_i mapping a strategy profile to a value. Here Φ represents the strategy space.
- $Z_i \in \mathbb{R}$: is the information utility[2] of party P_i.

[2] The information utility captures subjective valuation of privacy information by a party. Hence, we do not treat it as a function.

– $\mathcal{A}$: Set of parties caught deviating.

A rational party's utility function evaluates the gains or losses incurred by the party because of its strategic choices. In addition to monetary considerations, parties are also curious to learn information about the bids of other parties, provided their own privacy is not compromised in the process. For this, we consider *information utility* Z_i similar to the notion used in [GKS22].

The monetary utility function of an individual party P_i from participating in our protocol Π is as follows (where D is the security deposit paid by the bidder for participation in the auction):

$$U_i = \begin{cases} v_i - b_s & \arg\max(\mathbf{b}) = i \text{ and } P_i \text{ doesn't deviate} \\ 0 & \arg\max(\mathbf{b}) \neq i \text{ and } P_i \text{ doesn't deviate} \\ -D & P_i \text{ deviates and gets caught} \\ \frac{D|\mathcal{A}|}{(n-|\mathcal{A}|)} & P_i \text{ doesn't deviate, } \mathcal{A} \text{ is set of parties caught} \\ & \text{ deviating or abort from protocol.} \end{cases} \tag{1}$$

In an ideal execution with a trusted party, each party P_i learns the second highest bid b_s and the identity of the winning party P_w. Let this information be valued at $z_i \in \mathbb{R}^+$ by P_i. By correctness of our protocol, the information utility realized by P_i by following the protocol honestly is z_i.
Let A be an arbitrary set of parties other than P_i i.e., $A \subseteq [n] \setminus \{i\}$. Let b_A represent the bids of parties in set A. Then P_i would be interested to learn about some function $f_i(b_A)$. On the other hand, being privacy-sensitive, P_i would be wary of some $j \neq i$ such that P_j learns $f_j(b_i)$ for some function f_j. We capture these considerations as below:

$$Z_i \begin{cases} = z_i & \text{if } P_i \text{ learns no more than output of auction protocol.} \\ > z_i & \text{if } P_i \text{ learns } f_i(b_A) \text{ (as described above)} \\ < 0 & \text{if } \exists j \neq i \text{ such that } P_j \text{ learns } f_j(b_i) \text{ for some function } f_j. \\ < z_i & \text{if protocol aborts.} \end{cases} \tag{2}$$

The negative information utility holds even if P_i has additionally learned about $f_i(b_A)$. This is to emphasise that parties do not prefer to lose their privacy even if it helps in learning about other parties' bids. We refer to such parties as *Privacy-sensitive* rational parties.

We further consider a dictionary order of utility for each party:

$$(U_i, Z_i) < (U_i', Z_i'), if \begin{cases} (U_i < U_i') \vee \\ \big((Z_i < Z_i') \wedge (U_i = U_i')\big) \end{cases} \tag{3}$$

This order captures the assumption regarding rational parties that they value monetary utility to be higher than the information utility.

4 Our protocol

We now present a description of our second price auction protocol Π, which does not require any auctioneer and guarantees full privacy in the presence of privacy-sensitive rational parties.

4.1 Notation

We use a fixed publicly known value of the security deposit amount, D, for all parties. This value of D can be, for instance, the reserve price of the auctioned item. We use as building blocks a maliciously secure OT protocol Π_{OT}, a secure commitment scheme Com and non-interactive zero-knowledge (NIZK) proofs. We use a group $\mathbb{G}$ of prime order q where the DDH assumption holds.

The protocol uses a Bulletin Board (BB) for all communication. We use the notation $A_{ij,k}$ to denote the message or random value A generated by the party P_i while interacting with the party P_k during the round j. The protocol proceeds in l rounds. In each round j, a party P_i computes the bit it contributes, denoted by d_{ij}, as per Anonymous Bidding Protocol (ABP). We denote the encoding of d_{ij} as B_{ij}. ABP and its bit encoding scheme are described in Appendix A. The protocol also uses a fixed maximum time τ that a party gets to post its message, beyond which the party is considered to have aborted the protocol. We use $[0]_{ij}, [1]_{ij}$ to denote encoding of bits 0 and 1 respectively by P_i during round j. $\xleftarrow{\$}$ denotes uniform sampling from a certain distribution, $\xleftarrow{pay}$ denotes payment and $\xleftarrow{write}$ denotes writing to BB.

4.2 Construction

We describe our second price auction protocol as a set of algorithms. During the setup phase (Algorithm 1), parties allocate their secret keys, construct commitments and publish the public keys and commitments to the BB.

Algorithm 1 Setup Phase

```
1: Each $P_i$:
2: $P_i(v_i, b_i)$ // Initialize
3: $Deposit \xleftarrow{pay} D$ // Register for auction
4: $\mathsf{pp} = (\mathbb{G}, q, g, h, l, \tau)$ // Receive public parameters
5: for $j = 1$ to $l$ do
6:     $x_{ij}, r_{ij} \xleftarrow{\$} \mathbb{Z}_q$
7:     // Secret keys: $x_{ij}$ for encoding 0 and $r_{ij}$ for encoding 1
8:     $X_{ij} = g^{x_{ij}}$ // Public key
9:     $a_{ij} \xleftarrow{\$} \mathbb{Z}_q$ // Randomness for commitment
10:    $c_{ij} = \mathsf{Com}(b_{ij}, a_{ij})$
11:
12:    $BB \xleftarrow{write} (X_{ij}, c_{ij})$
13:    $Y_{ij} = \dfrac{\prod_{k=1}^{i-1} X_{kj}}{\prod_{k=i+1}^{n} X_{kj}}$
14: end for
15: $state_i \leftarrow$ IN-RACE
```

Auction phase (Algorithm 2) runs for l rounds. During each round, parties use ABP to determine their contribution bit for the round and exchange encoded bits through OT. At the end of each round, parties compute the output bit.

Algorithm 2 Auction Phase

```
1: Phase runs for $j = 1$ to $l$ rounds.
2: Each $P_i$ receiving from $P_k$ through OT during round $j$:
3: if $state_i$ = IN-RACE then
4:     $d_{ij} = b_{ij}$
5: end if
6: if $state_i$ = WINNER then
7:     $d_{ij} = 1$
8: end if
9: if $state_i$ = NOT-IN-RACE then
10:    $d_{ij} = 0$
11: end if
12: if $d_{ij} = 0$ then
13:    $B_{ij} = Y_{ij}^{x_{ij}}$ // $[0]_{ij}$
14: else
15:    $B_{ij} = g^{r_{ij}}$ // $[1]_{ij}$
16: end if
17: OT Receiver:($P_i$)
18: $\alpha_{ij,k} = d_{ij}$,  $\beta_{ij,k} \xleftarrow{\$} \mathsf{R}$
19: $\mathsf{otr}^1_{ij,k} = \mathsf{OT.R}_1(\alpha_{ij,k}, \beta_{ij,k})$,  $BB \xleftarrow{write} \mathsf{otr}^1_{ij,k}$
20: //Write OT receiver randomness to BB for $P_i \leftarrow P_k$
21: OT Sender: ($P_k$)
22: $M^{(1)}_{kj,i} = B_{kj}$
23: $\omega_{kj,i} \xleftarrow{\$} \mathbb{Z}_q$,  $M^{(0)}_{kj,i} = g^{\omega_{kj,i}}$.
24: // 0-Token of $P_k \to P_i$
25: $\delta_{kj,i} \xleftarrow{\$} \mathbb{Z}_q$,  $\Omega_{kj,i} = \mathsf{Com}(\omega_{kj,i}\delta_{kj,i})$,  $BB \xleftarrow{write} \Omega_{kj,i}$
26: // Write commitment to 0-token to BB
27: $\mathsf{ots}_{kj,i} = \mathsf{OT.S}\left(\mathsf{otr}^1_{ij,k}, M^{(0)}_{kj,i}, M^{(1)}_{kj,i}, \gamma_{kj,i} \xleftarrow{\$} \mathsf{R}\right)$
28: $BB \xleftarrow{write} \mathsf{ots}_{kj,i}$ // OT sender randomness of $P_k \to P_i$
29: $P_i$ retrieves $P_k$'s message as $B_{kj,i} = \mathsf{OT.R}_2(\mathsf{ots}_{kj,i}, \beta_{ij,k})$.
30: DO Winner Discovery (Algorithm 4)
31: DO Write to BB (Algorithm 5)
32: DO Cheater Detection (Algorithm 3)
33: DO Compute Output (Algorithm 6)
```

Each party contributing a 1 during a round checks locally if it is the winner, as in Algorithm 4. The winner discovery happens oblivious to other parties. After a successful winner discovery, winner's behavior is different from other parties as can be seen in Algorithm 5.

After l rounds of auction phase, verification phase (Algorithm 7) is initiated to check if the parties had been honest in their computation. If any cheating is detected or if there is any timeout noticed, the protocol terminates. The aborted and cheating parties lose their deposits which is redistributed among the honest parties. Honest parties get back their deposits.

Algorithm 3 *Cheater Detection*

```
1: Each $P_i$ during round $j$ with $(d_{ij} = 1)$:
2: if $B_{kj,i} \neq B_{kj}$ then
3: // Bit-code received through OT doesn't match with one on BB.
4: // $P_i, P_k$ open OT randomness, commitment and secret key for round $j$
5:     $BB \overset{write}{\longleftarrow} (\beta_{ij,k}, a_{ij}, x_{ij}, r_{ij})$
6:     $BB \overset{write}{\longleftarrow} (\gamma_{kj,i}, a_{kj}, x_{kj}, r_{kj})$
7:     $\mathsf{otr}' \leftarrow \mathsf{OT.R}_1(1, \beta_{ij,k})$
8:     $M \overset{\$}{\longleftarrow} \mathbb{G}, \mathsf{ots}' \leftarrow \mathsf{OT.S}_1\left(\mathsf{otr}^1_{ij,k}, M, B_{kj}, \gamma_{kj,i}\right)$
9:     if $(\mathsf{otr} = \mathsf{otr}^1_{ij,k}) \bigwedge (\mathsf{ots}' \neq \mathsf{ots}_{kj,i})$ then
10:        $P_k$ forfeits deposit to $P_i$ // $P_k$ is cheater
11:    else
12:        $P_i$ forfeits deposit to $P_k$ // $P_i$ is cheater
13:    end if
14:    $state_i$ = TERMINATE
15: end if
16: Each $P_i$ during round $j$
17: if Messages not received from $P_k$ within $\tau$ units OR Messages received from $P_k$ are malformed then
18:    Declare timeout against $P_k$
19:    $P_k$ forfeits deposit to honest parties // $P_k$ is cheater
20:    $state_i$ = TERMINATE
21: end if
```

Algorithm 4 *Winner Discovery*

```
1: Each $P_i$ during round $j$ with $d_{ij} = 1$:
2: $B = \prod_{k \in [n], k \neq i} B_{kj} \cdot [0]_{ij}$
3: if $B = 1$ then
4:     $state_i$ = WINNER
5: end if
```

Remark 1: Our protocol assumes a unique highest bidder. If there are multiple highest bidders, then the computed output also happens to be the highest bid. In such a case, we can use arbitrary tie-breaking mechanisms to choose the winner. The chosen winner pays the computed price.
Remark 2: We would like to emphasize that the Winner resolution procedure in Algorithm 7.6 acts as a deterrent for any rational party from cheating. A rational party does not use this mechanism to learn about the bid of another. This follows from our assumption that each party values its monetary utility to be higher than the information utility.

We show that the above protocol correctly computes the second highest bid in Section 5. In the following section, We analyze the strategies of a party participating in the protocol. There, we show that the honest strategy is weakly dominant for any party P_k. We then prove that this protocol does not leak anything, other than the protocol output, to any party that does not deviate from the protocol in Section 6.4. Transactions related to security deposits are handled outside the protocol (see Section 2.1).

5 Correctness of protocol Π

In this section, we will show that the protocol specified in Section 4.2 correctly computes the second highest bid and identifies the highest bidder.

Lemma 1. *In round j of SPA, for $j \in [l]$, the computed bit b_{sj} is the OR of contributed bits from all parties i.e., $b_{sj} = \vee_{i=1}^{n} d_{ij}$, for $i \in [n]$. Consequently, the computed bit is $\max_i(d_{ij})$, for $j \in [l]$.*

Proof. This follows directly from the property of the encoding scheme (as described in Appendix A). If at least one of the parties P_i has contributed 1-bit code B_{ij}, the product $\prod_{i=1}^{n} B_{ij} \neq 1$ and thus computed output $b_{sj} = 1$. On the other hand, if B_{ij} is 0-bit code for all $i \in [n]$ then $\prod_{i=1}^{n} B_{ij} = 1$ and thus computed output $b_{sj} = 0$. Since the OR function outputs the maximum of the input bits, the computed bit is $\max_i(d_{ij})$, for $j \in [l]$. □

Algorithm 5 *Write to BB*

```
1: if $state = winner$ then
2:     // Winner $P_w$ with ($state_w$ = WINNER) during round $j$:
3:     $B = \prod_{k\in[n],k\neq w} B_{kj} \cdot [0]_{wj}$
4:     if $B = 1$ then
5:         $BB \overset{write}{\longleftarrow} [0]_{wj}$
6:     else
7:         $BB \overset{write}{\longleftarrow} [1]_{wj}$
8:     end if
9: else// Not a winner
10:     $BB \overset{write}{\longleftarrow} B_{ij}$
11: end if
```

Algorithm 6 *Compute Output*

```
1: Each $P_i$ during round $j$:
2:     $B = \prod_{k\in[n]} B_{kj}$
3: if $B = 1$ then
4:         $b_{sj} = 0$
5: else
6:         $b_{sj} = 1$
7: end if
8: if $b_{sj} = 1 \wedge d_{ij} = 0$ then
9:     $state_i$ = NOT-IN-RACE
10: end if
11: After $l$ rounds, $b_s = (b_{s1}||\cdots||b_{sl})$
```

Lemma 2. *In the protocol specified in Section 4.2, there is exactly one party (denoted by P_w) that enters post winning phase, when all the parties follow the protocol honestly and have distinct bids.*

Proof. Suppose there is no $j \in [l]$ such that j is the winning round for any party. Then, we have that for every j, it is not the case that exactly one party contributes 1-bit. Also, note that an honest party contributes 1-bit only if they have not yet lost the auction. Now, Let us denote by S_k, the set of all parties who have not yet lost in the auction till round k. Then we have $|S_k| \geq |S_{k+1}|$, $\forall k \in [l-1]$. And we have, since no party has reached the post winning phase, $S_l \geq 2$. But this implies that all the parties in S_l have the same bid, which is contrary to our assumption that no parties have the same bids.
Now, suppose more than two parties reach the post winning phase. Then, one of the following cases occurs:

- Both parties reach post winning phase in same round: In the protocol, a party reaches the post winning phase only if it is the only party that contributes 1-bit in the OT phase of that round. Hence, two parties cannot reach post winning phase in the same round.
- One party reaches post winning phase before other party: If one party reaches post winning phase, then it always contributes 1 in the OT phase in all the subsequent rounds. Hence, after a party has reached the post winning phase, no other party can be the only party contributing 1 in any subsequent rounds.

□

Theorem 1. *The protocol specified in Section 4.2 computes the second highest bid correctly when all parties follow the protocol.*

Proof. From Lemma 1, we have that the computed output during each round is the highest of bit contributed from the bidders. Moreover, recall that the unique winner in the protocol writes the same bit codes to BB as that of the second highest bidder. This prevents the second highest bidder from assuming that it is the winner and also ensures that the second highest bidder does not drop out of the race all through the protocol. Thus, during every round, the computed bit corresponds to that of the second highest bid. □

Theorem 2. *The protocol specified in Section 4.2 correctly identifies the highest bidder when all parties follow the protocol.*

Algorithm 7 Verification phase

```
1:  Winner Claim:
2:  For $P_w$ claiming auction: // write private keys used for encoding
3:      $BB \overset{write}{\longleftarrow} x_{wj}, \forall j \in [l]$, with $b_{sj} = 0$ and
4:      $BB \overset{write}{\longleftarrow} r_{wj}, \forall j \in [l]$, with $b_{sj} = 1$
5:
6:  Winner resolution:
7:  Each $P_i$ claiming auction, for all $j \in [l]$ and $k \in [n], k \neq i$:
8:      $BB \overset{write}{\longleftarrow} (\beta_{ij,k}, \gamma_{ij,k})$
9:      $BB \overset{write}{\longleftarrow} (c_{ij}, a_{ij})$
10: Public verification of the consistency of claimant's messages
11:
12: Proof of correct choice bit:
13: Each $P_i, i \neq w$, for all $j \in [l]$ with $b_{sj} = 0$ and $k \in [n], k \neq i$:
14:     $BB \overset{write}{\longleftarrow} M^{(0)}_{kj,i}$ // 0-token received from $P_k$
15:     $BB \overset{write}{\longleftarrow} (\omega_{ij,k}, \delta_{ij,k})$ // 0-token Commitments sent to $P_k$
16:
17: Proof of correct computation:
18: Each $P_i, i \neq w$, for all $j \in [l]$:
19:     $BB \overset{write}{\longleftarrow}$ NIZK proof for relation in equation 8 of Appendix B.
20:
21: Honest parties get back their deposit.
22: Parties failing to provide 0-token or accepting NIZK proof are considered cheating and forfeit deposits, which
    are distributed among the honest parties.
```

Proof. From Lemma 2 we have a unique winner in the protocol. During the winner discovery round, the winner alone has a 1 bit to contribute, while every other party has a 0 bit to contribute. Thus, the winner's bid is higher than all other bidders. □

6 Analysis of strategies of rational parties in the protocol

We now present how our SPA protocol ties cryptography and game theory together to give stronger guarantees than conventional cryptographic MPC style protocols. To analyse all strategies available to a party, we describe how a deviation in the strategy of a party affects the protocol and view of such a party.

At any point during the protocol, a party's action (contribution to protocol) is decided by the strategy and the partial transcript available (its view). To analyse these coherently while maintaining ease of reading, we divide the set of all the deviating actions into the following partitions:

– DS_1: Deviations by sending malformed messages: These are detected, and senders lose their deposit.

– DS_2: Deviations in actions listed in Table 1:

  We use mix of rationality and cryptographic arguments to show these deviations are weakly dominated by honest actions.

– DS_{3a}: Deviations that are not part of Table 1, but leads to aborting of protocol (e.g. false winner claim, providing invalid proofs of computation etc). Here, we use various cryptographic constructs to detect the cheating.

– DS_{3b}: All other actions that do not affect the correctness of protocol but are deviations that can affect the privacy of the deviating party. Here, we use the privacy sensitivity of parties to argue that it is not rational for parties to deviate.

For the DS_1, DS_2, deviations have to occur in the Auction phase (Algorithm 2). The parties go through the auction phase in l-rounds, where each round comprises of three parts, exchanging encoded bits through OT, local computation to update state, and writing to BB. Correspondingly, in each round of protocol, every party P_i with strategy π_i contributes three types of input, namely, $\mathsf{cb}_{ij,k}$, $\mathsf{bcot}_{ij,k}$ and bcbb_{ij}. Here $\mathsf{cb}_{ij,k}$ denotes the choice bit used by the P_i as an OT receiver with P_k in round j of protocol; bcot_{ij}, k denotes the encoding of d_{ij} that P_i uses as OT sender in round j with P_k, and bcbb_{ij} is the encoding of d_{ij} that P_i writes on BB in round j. There is a unique honest strategy for each party. We describe all other strategies of a party as deviations from the honest strategy.

For the DS_{3a}, we consider deviations that are other than changing the contributing bit in a round. These deviations include false winner claim, and deviations in the proofs sought by the protocol. Here, we argue that such deviations are protected by the security of cryptographic primitives.

The DS_{3b} only comprises of deviations that are not included in the above two cases. For such deviations, verification goes through (protocol does not abort), and there is no deviation as per Table-1. In this case, protocol correctness is not affected, and so we argue such deviations can not increase the utility of the deviating party.

We denote the action selected by P_i in round j using honest strategy π_i while interacting with P_k as $s_{ij,k}$ and action selected using the deviating strategy as $s'_{ij,k}$. We also denote the winner discovery round by $\bar{j}$. For example, suppose P_i is interacting with P_k during round j. Then an action $t_{a\to b} \in \{\mathsf{cb}_{ij,k}, \mathsf{bcot}_{ij,k}, \mathsf{bcbb}_{ij}\}$, $a, b \in \{0.1\}$ means P_i is supposed to use the bit a in honest run of protocol, but chooses the bit b during action t. In each case, we show that the parties realize the best utility by following the protocol honestly.

	Actions using honest strategy	**Actions using dishonest strategy**
Choice bit selection	$\mathsf{cb}_{1\to1}, \mathsf{cb}_{0\to0}$	$\mathsf{cb}_{1\to0}, \mathsf{cb}_{0\to1}$
Bit code sent to other parties using OT	$\mathsf{bcot}_{1\to1}, \mathsf{bcot}_{0\to0}$	$\mathsf{bcot}_{1\to0}, \mathsf{bcot}_{0\to1}$
Bit codes written to BB	$\mathsf{bcbb}_{1\to1}, \mathsf{bcbb}_{0\to0}$	$\mathsf{bcbb}_{1\to0}, \mathsf{bcbb}_{0\to1}$

Table 1: Actions available for parties.

Remark 3: Only the highest bidder can come forward to claim the auction by paying the computed output. This is because, for any other bidder participating in a second price auction, it is not rational to pay the second highest bid.

Remark 4: In the following, where we show that honest strategies dominate over the deviations, we assume PPT rational parties. Hence, the inequalities (for the utilities of weakly dominant strategy) hold with an overwhelming probability. In other words, there exists only a negligible probability by which the parties can make gains with deviation.
We first prove a result that would be useful for subsequent proofs.

Lemma 3. *Let P_k be an arbitrary party who uses actions $\mathsf{bcot}_{a\to b}$ and $\mathsf{bcbb}_{a\to d}$ such that $b \neq d$ during some round j. Let P_i be the honest party who uses $d_{ij} = 1$. Assuming the correctness of Π_{OT} protocol, P_i will successfully identify P_k's cheating with overwhelming probability.*

Proof. Consider a cheating party P_k who uses actions $\mathsf{bcot}_{a\to b}$ and $\mathsf{bcbb}_{a\to d}$ such that $b \neq d$ during some round j. Then the party P_i with $d_{ij} = 1$ can show that P_k has indeed sent some bit code during OT and written another bit code to BB as described in step 3 cheater detection part of the protocol specification.

Firstly we claim that no cheating party P_i can implicate an honest party P_k. Suppose, on the contrary, that P_i claims P_k is cheating. In response, honest P_k successfully opens its OT sender randomness $\gamma_{kj,i}$ such that $\mathsf{ots}_{kj,i} = \mathsf{OT.S}\left(\mathsf{otr}^1_{ij,k}, M^{(0)}_{kj,i}, B_{kj}, \gamma_{kj,i}\right)$. Now P_i has to produce a valid bit code B'_{kj} to have been received from P_k through OT, such that,

$$B'_{kj} = \mathsf{OT.R}_2(\mathsf{ots}_{kj,i}, \beta'_{ij,k})$$
$$\implies \mathsf{ots}_{kj,i} = \mathsf{OT.S}\left(\mathsf{otr}^1_{ij,k}, M^{(0)}_{kj,i}, B'_{kj}, \gamma_{kj,i}\right)$$

However, this would contradict the correctness of Π_{OT}.

Likewise, cheating P_k can not refute honest P_i's claim by producing different sender randomness to show that the two bit codes are the same. Again, suppose on the contrary that P_k is able to produce two different OT sender randomness $\gamma_{kj,i}, \gamma'_{kj,i} \in \mathsf{R}$ such that

$$\mathsf{ots}_{kj,i} = \mathsf{OT.S}\left(\mathsf{otr}^1_{ij,k}, M^{(0)}_{kj,i}, B_{kj}, \gamma_{kj,i}\right)$$
$$= \mathsf{OT.S}\left(\mathsf{otr}^1_{ij,k}, M^{(0)}_{kj,i}, B'_{kj}, \gamma'_{kj,i}\right)$$

This means OT sender P_k is able to produce two different OT sender randomness, both of which generate the same OT sender message $\mathsf{ots}_{kj,i}$ for two different messages B_{kj}, B'_{kj} – contradicting the correctness of Π_{OT}. □

Lemma 4. *If no party in the protocol deviates corresponding to actions in Table 1 and the protocol does not abort, then the protocol computes the correct output.*

Proof. Suppose the protocol does not abort but outputs a wrong winning price. Note that the winning price is only a function of bcbb in the protocol. But since there is no deviation in bcbb by assumption, the winning price is computed correctly. If any other person comes forward or the winner does not come forward, then protocol leads to abort. □

Theorem 3. *Let $P_i, i \in [n]$ be any PPT party participating in the second price auction protocol Π described in Section 4.2. Then, assuming the security of the bit encoding scheme,* Com *is a secure commitment scheme and Π_{OT} is a malicious secure OT implementation, the protocol Π is a weakly dominant strategy equilibrium as per Definition 5.*

Proof. Consider an arbitrary rational PPT party P_i. Let π_i be the honest strategy (of following the protocol Π), and π_i' be the strategy in which P_i deviates during some round j.

We will now show that deviations as per the partitions DS_1, DS_2, DS_3, DS_4 are dominated by honest actions.

1. DS_1: This is the case where there exists a party that deviates by sending malformed messages during their protocol interactions. However, as per Algorithm 3, line 17, such actions result in protocol abort, and parties sending such messages lose their deposit. Thus, these actions are dominated by honest actions.
2. DS_2: This is the case where there exists a party that deviates with respect to actions in Table 1. From Lemmas 5, 6 and 7, we have that these deviations are dominated by the honest strategy for all parties.
3. DS_3: This is the case where no party deviates with respect to actions in Table 1
   (a) DS_{3a} : There exists a party that deviates in a way that leads to protocol abort. This happens when one or more parties are caught cheating as per Algorithms 3, 7. In each case, the deviating parties lose their deposit. Thus, the honest strategy π_i dominates over the deviating strategy for each P_i.
   (b) DS_{3b} : This is the case where protocol does not output abort. The protocol completes with correct output (by Lemma 4). Thus, there is no change in monetary utility for any party (monetary utility is a function of protocol output). The only thing such a deviation can hope to achieve is to gain information utility. We argue this does not happen as follows:
      – Learn information about other parties: A careful scrutiny of the protocol shows that the only actions to be considered in this case are when the party samples its private randomness improperly (or deviates in sampling a well-formed message that induces an improper distribution of randomness). We show what information a party P_i can learn about other parties with help of such a deviation. Other parties' messages are unaffected by such a deviation by P_i (other parties' messages are only a function of d_{ij} that they derive since there is no change as per Table-1, d_{ij} is derived correctly by other parties). Thus, P_i cannot get any more information than it would have gained while being honest.
      – Effects on one's own privacy as a result of such deviation: If the induced distribution of randomness remains uniform, then no deviation has occurred. In case the induced randomness is different from uniform, such deviation can potentially lead to loss of information utility of the deviating party as the private inputs of such a party become vulnerable – since the commitments may not be hiding and security of bit encoding scheme may not hold. Thus, for all privacy sensitive P_i, deviating actions are dominated by honest actions.

   Thus, for all privacy sensitive P_i, deviating actions are dominated by honest actions.

In case (3), where no party deviates in actions corresponding to Table 1, party may choose a strategy that leads to abort (case (3a)) or no abort (case 3(b)) with certain probabilities. But such a strategy is weakly dominated by honest strategy because in no case does the party increase its utility and, in some cases, decrease its utility. Thus, for all protocol actions of Π, we have:

$$U_i(\pi_i', \hat{\pi}_{-i}) \leq U_i(\pi_i, \hat{\pi}_{-i}), \forall i \in [n]$$

It follows that Π is indeed a *Weakly Dominant Strategy Equilibrium* as per Definition 5. □

6.1 Choice bit selection

If *state* = IN-RACE, each party P_i uses bit d_{ij} derived from ABP as its choice bit during round j. If *state* = WINNER, the party chooses 1 as the choice bit, and if *state* = NOT-IN-RACE, the party chooses 0 as the choice bit. Let us denote a party's action by $\mathsf{cb}_{a \to b}$. A deviating party can choose the choice bit to learn bit codes from other parties. Thus each party has the following actions available: $\{\mathsf{cb}_{0\to0}, \mathsf{cb}_{1\to1}, \mathsf{cb}_{1\to0}, \mathsf{cb}_{0\to1}\}$ during a particular round for choice bit selection.

Lemma 5. *For any rational PPT party P_i, let π_i be the honest strategy, and π'_i be the strategy in which P_i deviates using either $\mathsf{cb}_{1\to0}$ or $\mathsf{cb}_{0\to1}$ in some arbitrary round j. Then, assuming that Π_{OT} is a maliciously secure OT protocol, the strategy π_i weakly dominates the strategy π'_i for all $i \in [n]$ during all rounds $1 \le j \le l$ as per the* Weakly dominant strategy *Definition 4.*

Proof. We first note that any strategy followed by P_i for choice bit selection only affects the view of P_i, and the view of all other parties remains unaffected. Suppose P_i uses a cheating strategy while interacting with another party P_k. I.e., $s_{ij,k} \in \{\mathsf{cb}_{1\to0}, \mathsf{cb}_{0\to1}\}$, for arbitrary i, j and k with $i \neq k$. The following cases can occur:

- $\mathsf{cb}_{1\to0}$: Suppose $j < \bar{j}$, where $\bar{j}$ is the "winning round". Then, P_i will not be able to perform winner discovery (Algorithm 7.4) as it has not received the correct bit codes. If P_i can perform winner discovery, that would require P_i to come up with encoded bits of other parties, and in such a case, we can construct an adversary who can break the OT sender security of Π_{OT} protocol. In this case, the following can occur:
  - If P_i does not cheat in $\mathsf{bcot}_{ij,k}$ and bcbb_{ij}, then there is no change in view of other parties, and this does not affect the protocol. In this case, there is no change in gain in the utility of P_i.
  - If P_i cheats in $\mathsf{bcot}_{ij,k}$ and/or bcbb_{ij}, then we will show in the next lemmas that there is a loss in the utility of P_i.

  On the other hand, if $j \ge \bar{j}$, P_i will not be able to execute its honest strategy in round j while writing to BB. Because P_i can not compute the second highest bidder's contribution during winner discovery Algorithm 4.2 as P_i cannot recover the contribution of P_k. If it could, we can construct an adversary who can break the OT sender security of Π_{OT} protocol. Therefore, with non-zero probability, P_i cheats in $\mathsf{bcot}_{ij,k}$ and/or bcbb_{ij} and ends up losing deposit. In both the above cases, P_i's utility is no better than staying honest, i.e., $\forall i \in [n], k \neq i$:

  $$U_i(\pi'_i, \hat{\pi}_{-i})_{s'_{ij,k}=\mathsf{cb}_{1\to0}} \le U_i(\pi_i, \hat{\pi}_{-i})_{s_{ij,k}=\mathsf{cb}_{1\to1}}$$

- $\mathsf{cb}_{0\to1}$: In this case, observe that P_i will be caught if P_i does not claim the auction, as P_i will not be able to provide 0-token for round j when $b_{sj} = 0$ (Algorithm 7. 12). If it could, we can construct an adversary who can break the OT sender security of Π_{OT} protocol. . On the other hand, if $b_{sj} = 1$, it doesn't learn any extra information than while being honest. Thus, $\forall i \in [n], k \neq i$:

  $$U_i(\pi'_i, \hat{\pi}_{-i})_{s'_{ij,k}=\mathsf{cb}_{0\to1}} \le U_i(\pi_i, \hat{\pi}_{-i})_{s_{ij,k}=\mathsf{cb}_{0\to0}}$$

  So, let us consider the case when P_i claims the auction. Following are the two cases that can occur:
  - P_i is the actual winner: In this case $d_{kj} = 0, \forall k \neq i$. Since P_i would have anyway learned the encoding of other parties' bits from the BB round, P_i does not gain any information utility in this case.
  - P_i is not the actual winner: In this case, there is a player P_w with a bid higher than P_i and $w \neq i$. The following cases can occur:
    - ∗ Protocol output (price of the auction) $< b_w$: In this case, P_w will also claim the auction if it is honest. P_i's cheating will be caught during winner resolution (Algorithm 7.6). Hence the monetary utility of P_i reduces in this case.
    - ∗ Protocol output (price of the auction) $\ge b_w$: In this case, P_i has to pay more than b_i and hence its monetary utility decreases.

  From the above cases, we have $\forall i \in [n], k \neq i$:

  $$U_i(\pi'_i, \hat{\pi}_{-i})_{s'_{ij,k}=\mathsf{cb}_{0\to1}} \le U_i(\pi_i, \hat{\pi}_{-i})_{s_{ij,k}=\mathsf{cb}_{0\to0}}$$

To argue that honestly following the protocol is a weakly dominant strategy, consider the strategy set $\hat{\pi}_{-i}$ wherein at least one other party P_k deviates using the action $\mathsf{cb}_{0\to 1}$ during every round, i.e., $s'_{kj,i} = \mathsf{cb}_{0\to 1}, \forall j \in [l], i \neq k$. In this case, if P_i chooses to be honest, P_i is guaranteed to get the security deposit redistribution of a cheater. On the other hand, P_i's deviation would result in either losing the auction or getting caught to lose the security deposit. Thus in this case,

$$U_i(\pi'_i, \hat{\pi}_{-i}) < U_i(\pi_i, \hat{\pi}_{-i}), \forall i \in [n]$$

Thus P_i's weakly dominant strategy is to stay honest and use actions $s_{ij,k} \in \{\mathsf{cb}_{0\to 0}, \mathsf{cb}_{1\to 1}\}, \forall k \in [n]$ for all rounds $j \in [l]$, irrespective of the strategies $\hat{\pi}_{-i}$ of other parties. Thus we have,

$$U_i(\pi'_i, \hat{\pi}_{-i}) \leq U_i(\pi_i, \hat{\pi}_{-i}), \forall i \in [n]$$

□

6.2 Bit codes sent to other parties using OT

Each party shares its bit codes through OT to all other parties during each round. Only parties using a 1 as the choice bit can retrieve these bit codes for their local computation. We denote this action by $\mathsf{bcot}_{a\to b}$. A deviating party can choose to send bit codes so as to learn the highest bid. Thus, each party has the following actions available to it: $\{\mathsf{bcot}_{0\to 0}, \mathsf{bcot}_{1\to 1}, \mathsf{bcot}_{1\to 0}, \mathsf{bcot}_{0\to 1}\}$ during each round.

Lemma 6. *For any rational PPT party P_i, let π_i be the honest strategy, and π'_i be the strategy in which P_i deviates using either $\mathsf{bcot}_{1\to 0}$ or $\mathsf{bcot}_{0\to 1}$ in some round. Then, assuming the security of the encoding scheme, Π_{OT} is a maliciously secure OT protocol and NIZK proofs are sound, the strategy π_i weakly dominates the strategy π'_i for all $i \in [n]$ during all rounds as per the* Weakly dominant strategy *Definition 4.*

Proof. Consider an arbitrary round j. Party P_i does not have any information about d_{kj} for any $k \neq i$ before the OT round. Therefore, the choice of $\mathsf{bcot}_{ij,k}$ is independent of d_{kj}. Now, the following cases can arise:

- $d_{kj} = 0$: In this case, if P_k is honest in $\mathsf{cb}_{kj,i}$ selection, then choice of $\mathsf{bcot}_{ij,k}$ makes no difference in the protocol. Hence, deviating P_i does not gain any utility in this case. On the other hand, if P_k cheats, P_k will not be able to produce 0-token for the round and will get caught (Algorithm 7.12). Else, we can construct an adversary who can break the sender security of Π_{OT} protocol. In such a case, P_i would get the redistribution of the security deposit if it does not get caught (also by staying honest) and loses the security deposit if gets caught. In either case, P_i's utility does not increase by cheating.
- $d_{kj} = 1$: In this case, if P_k chooses to cheat in $\mathsf{cb}_{kj,i}$ selection, then P_i's utility remains at most as much as its utility without cheating; but in case P_k is honest in the selection of $\mathsf{cb}_{kj,i}$, then by Lemma 3, P_k can catch the cheating behavior of P_i if P_i does not cheat in bcbb_{ij}. If P_i cheats in bcbb_{ij}, then P_i would fail to provide accepting NIZK proof (Algorithm 7.17). This is because if P_i can provide the accepting NIZK proof, one can construct an adversary who can break the soundness of the NIZK proof. Either way $\forall i \in [n], k \neq i$ and $a \neq b$,

$$U_i(\pi'_i, \hat{\pi}_{-i})_{(s_{ij,k}=\mathsf{bcot}_{a\to b})} \leq U_i(\pi_i, \hat{\pi}_{-i})_{(s_{ij,k}=\mathsf{bcot}_{a\to a})}$$

Lastly, consider the strategy set $\hat{\pi}_{-i}$ wherein at least one other party P_k uses the action $\mathsf{bcot}_{0\to 1}$ during every round, i.e., $s'_{kj,i} = \mathsf{bcot}_{0\to 1}, \forall j \in [l], i \neq k$. In this case, if P_i chooses to be honest, P_i is guaranteed to get the security deposit redistribution of a cheater. On the other hand, P_i's deviation would result in either losing the auction or getting caught to lose the security deposit. Thus in this case,

$$U_i(\pi'_i, \hat{\pi}_{-i}) < U_i(\pi_i, \hat{\pi}_{-i}), \forall i \in [n]$$

Thus P_i's weakly dominant strategy is to stay honest and choose $s_{ij,k} \in \{\mathsf{bcot}_{0\to 0}, \mathsf{bcot}_{1\to 1}\}$ for all rounds $j \in [l]$ while interacting with all parties $k \in [n], k \neq i$, irrespective of the strategies $\hat{\pi}_{-i}$ of other parties. Hence we have,

$$U_i(\pi'_i, \hat{\pi}_{-i}) \leq U_i(\pi_i, \hat{\pi}_{-i}), \forall i \in [n]$$

□

6.3 Bit codes written to BB

Towards the end of each round, every party writes the bit codes used for computation during that round onto BB. This is to facilitate the computation of the auction-price bit for the round. The actions chosen by a party for this step are denoted by $\mathsf{bcbb}_{a \to b}$ where $a, b \in \{0, 1\}$. Since each party evaluates the computed bit using bit codes on BB to decide whether it stays in the race or not, these actions are decisive. Thus each party has the following actions available to it: $\{\mathsf{bcbb}_{0\to0}, \mathsf{bcbb}_{1\to1}, \mathsf{bcbb}_{1\to0}, \mathsf{bcbb}_{0\to1}\}$ during each round. We would like to emphasize that no bidder can use the action $\mathsf{bcbb}_{0\to1}$ to become a winner. This is because writing to the BB does not affect winning chances but only affects the price to be paid for the auction. Moreover, deviating actions $\mathsf{bcbb}_{0\to1}, \mathsf{bcbb}_{1\to0}$ do not help to fetch any extra information than while being honest.

Lemma 7. *For any rational PPT party P_i, let π_i be the honest strategy, and π'_i be the strategy in which P_i deviates using either $\mathsf{bcbb}_{1\to0}$ or $\mathsf{bcbb}_{0\to1}$ in some round. Then, assuming the security of the bit encoding scheme and NIZK proofs are sound, the strategy π_i weakly dominates the strategy π'_i for all $i \in [n]$ during all rounds as per the* Weakly dominant strategy *Definition 4.*

Proof. Note that any party P_i that cheats in bcbb_{ij} and does not claim the auction gets caught because they cannot provide an accepting NIZK proof (Algorithm 7.17). This follows because, otherwise, one can construct a NIZK adversary who can break the soundness of NIZK proofs. Thus,

$$U_i(\pi'_i, \hat{\pi}_{-i}) \leq U_i(\pi_i, \hat{\pi}_{-i}), \forall i \in [n]$$

So the only case to be considered is if P_i cheats in bcbb_{ij} and claims the auction. The following cases can occur:

- $\mathsf{bcbb}_{0\to1}$: In this case, the computed value (the auction price) is more than P_i's bid. Hence P_i will lose monetary utility $\forall i \in [n], k \neq i$:

$$U_i(\pi'_i, \hat{\pi}_{-i})_{(s_{ij,k}=\mathsf{bcbb}_{0\to1})} < U_i(\pi_i, \hat{\pi}_{-i})_{(s_{ij,k}=\mathsf{bcbb}_{0\to0})}$$

- $\mathsf{bcbb}_{1\to0}$: Firstly, observe that this case can occur only when $j \neq \bar{j}$. However, in this case, there exists at least another party P_k with $d_{kj} = 1$. If P_k is honest, the computed output will be 1, and hence P_i would not be able to provide winning proof due to the security of bit encoding scheme (Algorithm 7.1) and loses deposit. If P_k is dishonest, P_i's utility would be no better than staying honest. Hence $\forall i \in [n], k \neq i$:

$$U_i(\pi'_i, \hat{\pi}_{-i})_{(s_{ij,k}=\mathsf{bcbb}_{1\to0})} \leq U_i(\pi_i, \hat{\pi}_{-i})_{(s_{ij,k}=\mathsf{bcbb}_{1\to1})}$$

Thus P_i's weakly dominant strategy is to honestly choose $s_{ij,k} \in \{\mathsf{bcbb}_{0\to0}, \mathsf{bcbb}_{1\to1}\}$, $\forall k \in [n]$ for all rounds $j \in [l]$, irrespective of the strategies $\hat{\pi}_{-i}$ of other parties. Thus for all $i \in [n]$ we have:

$$U_i(\pi'_i, \hat{\pi}_{-i}) \leq U_i(\pi_i, \hat{\pi}_{-i})$$

□

6.4 Privacy preserving computational dominant strategy equilibrium

Having established the equilibrium from Theorem 3, to argue privacy, it suffices to show that nothing beyond the output is learned by parties who are non-deviating. For this, we use the *ideal world – real world paradigm.* First, we present the description of a simulator for parties in different roles (winner, second highest bidder and others). We would like to emphasize that due to the symmetric nature of our protocol, it suffices to show the simulator for one party, which would be the same for others, too. We then show that the view of a party P_k in a real run of the protocol is indistinguishable from the output of the simulator. We denote by View_k^{Π}, the distribution of the view of party P_k while participating in protocol Π.

Theorem 4. *Assuming security of bit encoding scheme,* Com *is a secure commitment scheme, Π_{OT} is a semi-honest secure OT protocol, and zero knowledge property holds for the NIZK proofs, Π specified in Section 4.2 securely realizes the functionality $\mathcal{F}$ in the presence of non-deviating parties.*

Proof. **Ideal Functionality.** We consider the ideal functionality $\mathcal{F}$ to identify the winner and to compute the second highest bid. The functionality $\mathcal{F}$ operates with a set of parties $P = \{P_1, \ldots, P_n\}$ as follows:

- Each party P_i invokes $\mathcal{F}$ with input b_i.
- Once all n bids are available, $\mathcal{F}$ computes:
  $w = \arg\max_i(b_i)$ and $b_s = \max_{i \neq w}(b_i)$
- $\mathcal{F}$ returns the tuple (w, b_s) to each party.

  We will construct a simulator $\mathcal{S}$ for a PPT semi-honest party P_k, such that

$$\mathsf{View}_k^{\Pi} \approx_c \mathcal{S}_{P_k}^{\mathcal{F}}$$

  The simulator is invoked with public parameters pp, and bid b_k of P_k.

Setup Phase:

- $\mathcal{S}$ samples secret keys and randomness $x_{ij}, r_{ij} \xleftarrow{\$} \mathbb{Z}_q$ on behalf of all parties $P_i, i \in [n]$. It samples OT sender randomness $\gamma_{ij,m} \in \mathsf{R}$ and OT receiver randomness $\beta_{ij,m} \in \mathsf{R}$ for all P_i interacting with P_m during round $j \in [l]$. Computes the public keys $X_{ij} = g^{x_{ij}}, \forall i \in [n]$.
- $\mathcal{S}$ generates the 0-tokens $\omega_{ij,m}$ for each party P_i interacting with party P_m, for rounds $1 \leq j \leq l$.
- $\mathcal{S}$ generates the corresponding commitments to 0-tokens: chooses $\delta_{ij,m} \xleftarrow{\$} \mathbb{Z}_q, j \in [l]$, computes $\Omega_{ij,m} = g^{\omega_{ij,m}} h^{\delta_{ij,m}}$.
- For each party $P_i, i \neq k, i \neq s, i \neq w$: $\mathcal{S}$ constructs commitments to bits of bids of 0: chooses $a_{ij} \xleftarrow{\$} \mathbb{Z}_q, j \in [l]$, computes $c_{ij} = g^0 h^{a_{ij}}$.
- $\mathcal{S}$ invokes $\mathcal{F}$ on behalf of P_k with input b_k to obtain (w, b_s).
- If $b_k \neq b_s$, one of the parties is designated as P_s. For this party, $\mathcal{S}$ constructs commitments $c_{sj} = \mathsf{Com}(b_{sj}, a_{sj})$. Else commitments are constructed for bits of b_k.
  If $k \neq w$, $\mathcal{S}$ chooses $\bar{j}$ to be the smallest value of j such that $b_{sj} = 0, 1 \leq j \leq l$. $\mathcal{S}$ chooses b_w such that its first $\bar{j} - 1$ bits are equal to corresponding bits of b_s, $b_{w\bar{j}} = 1$ and remaining bits are all 0. $\mathcal{S}$ computes bit commitments for party P_w for bits b_{wj} and party P_k for bits $b_{kj}, j \in [l]$.
  If $k = w$, $\mathcal{S}$ computes bit commitments for party P_k for bits $b_{kj}, j \in [l]$.
  All commitments are written to BB.

Auction phase: For this phase, we need to distinguish three cases.

1. P_k is the winner :
   - In this case, $b_w = b_k$. $\mathcal{S}$ identifies the round $\bar{j}$ in which b_w differs from b_s.
   - For $1 \leq j < \bar{j}$ and each $i \neq k$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(b_{kj}, \beta_{kj,i})$ to obtain $\mathsf{otr}^1_{kj,i}$. For $\bar{j} \leq j \leq l$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(1, \beta_{kj,i})$ to obtain $\mathsf{otr}^1_{kj,i}$.
   - For the party P_s, $1 \leq j \leq l$ and $i \neq s$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(b_{sj}, \beta_{sj,i})$ to obtain $\mathsf{otr}^1_{sj,i}$.
   - For all other parties P_m, for $1 \leq j \leq l$ and $i \neq m$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(0, \beta_{mj,i})$ to obtain $\mathsf{otr}^1_{mj,i}$.
     All otr^1 messages are written to BB.
   - For the winner, $\mathcal{S}$ generates B_{wj} for the bits corresponding to b_s for all rounds $1 \leq j \leq l$. $\mathcal{S}$ generates bit codes B_{sj} for the bits corresponding to b_s for all rounds $1 \leq j \leq l$. For the remaining bidders, $\mathcal{S}$ generates 0-bit codes for all rounds. All bit codes are written to BB at the end of each round.
   - For P_w, $\mathcal{S}$ computes
     $\mathsf{OT.S}(\mathsf{otr}^1_{ij,w}, g^{\omega_{wj,i}}, B_{wj}, \gamma_{wj,i})$ to obtain $\mathsf{ots}_{wj,i}$ for all parties $i \neq w$ and for $1 \leq j < \bar{j}$. For rounds $\bar{j} \leq j \leq l$, $\mathcal{S}$ samples $M'_{wj} \xleftarrow{\$} \mathbb{G}$ and runs $\mathsf{OT.S}(\mathsf{otr}^1_{ij,w}, g^{\omega_{wj,i}}, M'_{wj}, \gamma_{wj,i})$ to obtain $\mathsf{ots}_{wj,i}$ for all parties $i \neq w$.
   - For P_s, using $\gamma_{sj,i}$, $\mathcal{S}$ invokes $\mathsf{OT.S}(\mathsf{otr}^1_{ij,s}, g^{\omega_{sj,i}}, B_{sj}, \gamma_{sj,i})$ to obtain $\mathsf{ots}_{sj,i}$ for each OT receiver P_i.
   - For P_m where $m \neq w, m \neq s$, using $\gamma_{mj,i}$, $\mathcal{S}$ runs $\mathsf{OT.S}(\mathsf{otr}^1_{ij,m}, g^{\omega_{mj,i}}, B_{mj}, \gamma_{mj,i})$ to obtain $\mathsf{ots}_{mj,i}$ for each OT receiver P_i.
     All ots messages are written to BB.
2. P_k is the second highest bidder:
   - In this case, the input $b_k = b_s$. $\mathcal{S}$ chooses $\bar{j}$ to be the smallest value of j such that $b_{sj} = 0$.
   - For $1 \leq j < \bar{j}$ and $i \neq w$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(b_{sj}, \beta_{wj,i})$ to obtain $\mathsf{otr}^1_{wj,i}$ for OT sender P_i. For $\bar{j} \leq j \leq l$ and $i \neq w$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(1, \beta_{wj,i})$ to obtain $\mathsf{otr}^1_{wj,i}$.

– For the party P_k, $1 \le j \le l$ and $i \neq s$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(b_{sj}, \beta_{sj,i})$ to obtain $\mathsf{otr}^1_{sj,i}$.

– For all other parties P_m $m \neq w, m \neq s$, for $1 \le j \le l$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(0, \beta_{mj,i})$ to obtain $\mathsf{otr}^1_{mj,i}$. All otr^1 messages are written to BB.

– For the winner P_w, $\mathcal{S}$ generates B_{wj} for the bits corresponding to b_{sj} for all rounds. $\mathcal{S}$ generates bit codes B_{sj} for the bits corresponding to b_s for all rounds. For the remaining bidders, $\mathcal{S}$ generates 0-bit codes for all rounds. All bit codes are written to BB at the end of each round.

– For P_w, $\mathcal{S}$ computes $\mathsf{OT.S}(\mathsf{otr}^1_{ij,w}, g^{\omega_{wj,i}}, B_{wj}, \gamma_{wj,i})$ to obtain $\mathsf{ots}_{wj,i}$ for all parties $i \neq w$ and for $1 \le j < \bar{j}$. For rounds $\bar{j} \le j \le l$, $\mathcal{S}$ samples $M'_{wj} \xleftarrow{\$} \mathbb{G}$ and runs $\mathsf{OT.S}(\mathsf{otr}^1_{ij,w}, g^{\omega_{wj,i}}, M'_{wj}, \gamma_{wj,i})$ to obtain $\mathsf{ots}_{wj,i}$ for all parties $i \neq w$.

– For P_s, $\mathcal{S}$ invokes $\mathsf{OT.S}(\mathsf{otr}^1_{ij,s}, g^{\omega_{sj,i}}, B_{sj}, \gamma_{sj,i})$ to obtain $\mathsf{ots}_{sj,i}$ for each OT receiver $P_i, i \neq s$.

– For P_m where $m \neq w, m \neq s$, $\mathcal{S}$ runs $\mathsf{OT.S}(\mathsf{otr}^1_{ij,m}, g^{\omega_{mj,i}}, B_{mj}, \gamma_{mj,i})$ to obtain $\mathsf{ots}_{mj,i}$ for each OT receiver $P_i, i \neq m$.
All ots messages are written to BB.

3. P_k is neither winner nor second highest bidder.

– In this case $b_k \neq b_s, b_k \neq b_w$. $\mathcal{S}$ chooses $\bar{j}$ to be the smallest value of j such that $b_{sj} = 0, 1 \le j \le l$.

– $\mathcal{S}$ compares the value b_k with b_s and identifies the round when P_k would drop out of race and computes the values of d_{kj} for $1 \le j \le l$.

– For $1 \le j < \bar{j}$ and $i \neq w$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(b_{sj}, \beta_{wj,i})$ to obtain $\mathsf{otr}^1_{wj,i}$. For $\bar{j} \le j \le l$ and $i \neq w$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(1, \beta_{wj,i})$ to obtain $\mathsf{otr}^1_{wj,i}$.

– For the party P_s, $1 \le j \le l$ and $i \neq s$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(b_{sj}, \beta_{sj,i})$ to obtain $\mathsf{otr}^1_{sj,i}$.

– For the party P_k, $1 \le j \le l$ and $i \neq k$, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(d_{kj}, \beta_{kj,i})$ to obtain $\mathsf{otr}^1_{kj,i}$.

– For all other parties P_m $m \neq w, m \neq s, m \neq k$, for $1 \le j \le l$ and for all other P_i, $\mathcal{S}$ invokes $\mathsf{OT.R}_1(0, \beta_{mj,i})$ to obtain $\mathsf{otr}^1_{mj,i}$.
All otr^1 messages are written to BB.

– For the winner P_w, $\mathcal{S}$ generates B_{wj} for the bits corresponding to b_s for all rounds $1 \le j \le l$. $\mathcal{S}$ generates bit codes B_{sj} for the bits corresponding to b_s for all rounds $1 \le j \le l$. Then $\mathcal{S}$ generates bit codes B_{kj} for the bits corresponding to d_{kj} for all rounds $1 \le j \le l$. For the remaining bidders, $\mathcal{S}$ generates 0-bit codes for all rounds. All bit codes are written to BB at the end of each round.

– For P_w, using $\gamma_{wj,i}$ $\mathcal{S}$ computes $\mathsf{OT.S}(\mathsf{otr}^1_{ij,w}, g^{\omega_{wj,i}}, B_{wj}, \gamma_{wj,i})$ to obtain $\mathsf{ots}_{wj,i}$ for all parties $i \neq w$ and for $1 \le j < \bar{j}$. For rounds $\bar{j} \le j \le l$, $\mathcal{S}$ samples $M'_{wj} \xleftarrow{\$} \mathbb{G}$ and runs $\mathsf{OT.S}(\mathsf{otr}^1_{ij,w}, g^{\omega_{wj,i}}, M'_{wj}, \gamma_{wj,i})$ to obtain $\mathsf{ots}_{wj,i}$ for all parties $i \neq w$.

– For P_s using $\gamma_{sj,i}$, $\mathcal{S}$ computes $\mathsf{OT.S}(\mathsf{otr}^1_{ij,s}, g^{\omega_{sj,i}}, B_{sj}, \gamma_{sj,i})$ to obtain $\mathsf{ots}_{sj,i}$ for each OT receiver P_i.

– For P_k using $\gamma_{kj,i}$, $\mathcal{S}$ computes $\mathsf{OT.S}(\mathsf{otr}^1_{ij,k}, g^{\omega_{kj,i}}, B_{kj}, \gamma_{kj,i})$ to obtain $\mathsf{ots}_{kj,i}$ for each OT receiver P_i.

– For P_m where $m \neq w, m \neq s, m \neq k$ using $\gamma_{mj,i}$, $\mathcal{S}$ runs $\mathsf{OT.S}(\mathsf{otr}^1_{ij,m}, g^{\omega_{mj,i}}, B_{ij}, \gamma_{mj,i})$ to obtain $\mathsf{ots}_{mj,i}$ for each OT receiver P_i.
All ots messages are written to BB.

For each of the above cases, during **Verification Phase**, $\mathcal{S}$ does the following:

– $\mathcal{S}$ generates NIZK proofs for all losing parties for the decider rounds as per their bids and writes to BB.

– $\mathcal{S}$ opens the private keys x_{wj}, r_{wj} used for encoding the bit codes of winner P_w for all rounds, $1 \le j \le l$ and writes them to BB.

– $\mathcal{S}$ provides the 0-tokens on behalf of all losing parties for the rounds when computed output is 0, and writes them to BB. Corresponding 0-token commitments for each party are also opened onto BB.

In order to argue indistinguishability, we consider the following hybrids.

- H_0: This is the real run of the protocol where parties generate actual public keys, bit commitments to the bits of their bids, bit codes corresponding to their bids, 0-tokens and their commitments, NIZK proofs and generate OT messages as per their contributed bits for the round.
- H_1: This is same as H_0 except for following. In the verification phase, NIZK proofs are replaced with simulated proofs by invoking the Zero Knowledge simulator that is guaranteed by the NIZK construction.
  By the zero knowledge property of NIZK proofs, H_1 is indistinguishable from H_0 for all PPT distinguishers.
- H_2: This is same as H_1 except for following. During the auction phase, $\beta_{ij,m} \in \mathsf{R}$ are sampled for each party P_i acting as OT receiver while interacting with OT sender P_m where $i \neq k, i \neq s, i \neq w$. $\mathsf{OT.R}_1(0, \beta_{ij,m})$ is used to compute $\mathsf{otr}^1_{ij,m}$ and written to BB.
  By receiver security of Π_{OT}, H_2 is indistinguishable from H_1 for all PPT distinguishers.
- H_3: This is same as H_2 except for following. During auction phase, for each OT sender P_i interacting with OT receiver P_m, $\mathsf{OT.S}(\mathsf{otr}^1_{mj,i}, g^{\omega_{ij,m}}, B_{ij}, \gamma_{ij,m})$ is invoked with B_{ij} being 0-bit code, to compute $\mathsf{ots}_{ij,m}$ for all parties $i \neq k, i \neq s, i \neq w$ and for $1 \leq j < l$ and written to BB. Moreover, at the end of the round same 0-bit codes B_{ij} are written to the BB.
  By security of the bit encoding scheme, H_3 is indistinguishable from H_2 for all PPT distinguishers.
- H_4: This is same as H_3 except for following. During the setup phase, commitments are for 0-bits for all parties $P_i, i \neq k, i \neq s, i \neq w$ and written to BB.
  By hiding property of the commitment scheme Com, H_4 is indistinguishable from H_3 for all PPT distinguishers.

Observe that the hybrid H_4 does not have any information about the bids of parties other than P_k and P_s. Thus P_k cannot learn anything about losing bids in H_4. Also, in H_0, the view of P_k corresponds to the real run, whereas H_4 corresponds to $\mathcal{S}$'s output. Moreover, by transitivity, $H_0 \approx_c H_4$. Thus, it follows that,

$$\mathsf{View}^{\Pi}_k \approx_c \mathcal{S}^{\mathcal{F}}_{P_k}$$

□

The following Theorem stating that Π is a *Privacy preserving computational dominant strategy equilibrium*, follows as a corollary of Theorems 3 and 4.

Theorem 5. *Let $P_i, i \in [n]$ be rational parties with respective utility functions $(U_1, \ldots, U_n)$ as described above. Assuming security of bit encoding scheme, Com is a secure commitment scheme and Π_{OT} is a maliciously secure OT protocol, and zero knowledge property holds for the NIZK proofs, the protocol Π described in Section 4.2 is a Privacy enhanced computational dominant strategy equilibrium as per Definition 6.*

6.5 Colluding parties

We consider bidders participating in the auction to be individually rational and can strategically act to enhance their utilities. One such action would be to collude with other bidders. There have been different approaches to model and tackle such collusion in the literature. Micali and Rabin [MR14] discuss the prevention of collusion among bidders to reduce the auction price. They propose rewarding the second highest bidder. As a result, parties would strive to be either highest or second highest bidders, thus stay truthful and prevent collusion among bidders. However, their scheme requires a central auctioneer and leaks the identity of the second highest bidder. Some works treat collusion to be managed by a single rational adversary [GKTZ12] while other parties are treated as honest. Other works demand that Nash equilibrium be achieved even if some of the parties deviate, referred to as $t-$resilient Nash equilibrium by Katz [Kat08] and m-stability in the work of Biçer, Yildiz and Küpçü [BYK21]. Instead, we treat the formation of collusion as a strategic move that rational parties choose. Our modelling has two important consequences: a) parties who are not part of collusion are not restricted, to be honest − but rational, strategically deciding to stay out of the collusion. b) As we will show in Lemma 8, this model helps to get collusion-resistance for free in our protocol.
In our setting, a group of parties come together to form a collusion if each of them is better off as a part of the group rather than acting alone. We believe that such a notion of collusion is more natural in a rational framework. Since each party joins a collusion for individual gains, the utility of a collusion inherently depends on utilities of individual players. In this work for auctions, where we model players as privacy-sensitive, we consider the utility of parties in a collusion as follows:

- Monetary utility: A collusion may choose to redistribute the total monetary utility gain in any arbitrary fashion. In such redistribution, no party should lose their monetary utility as compared to them acting alone.
- Information utility: Since colluding parties are individually rational, no such party prefers to lose any information utility by colluding. Hence, we define information utility of collusion to be positive only when no colluding party loses individual information utility. For this, we consider privacy-sensitive parties (a refinement of Equation 2) who value their privacy more than the information gained from learning about the private inputs of other parties. Such parties would not take up any action in the protocol that might result in leakage of their private inputs.

To summarize, we consider individually rational parties who do not form any collusion unless they can make gains from such a collusion and their private information is protected. Let P_i be an arbitrary privacy sensitive PPT rational party. Let $\mathcal{C}$ be an arbitrary collusion with $P_i \in \mathcal{C}$ and $\bar{\mathcal{C}}$ be set of parties not in $\mathcal{C}$. Let $b_{\bar{\mathcal{C}}}$ represent the bids of parties in set $\bar{\mathcal{C}}$. Then each $P_i \in \mathcal{C}$ would be interested to learn about some function $f_i(b_{\bar{\mathcal{C}}})$. On the other hand, being privacy-sensitive, P_i would be wary of some $j \neq i$ such that P_j learns $f_j(b_i)$ for some function f_j. These considerations are captured by modifying the information utility of Equation 2 as below:

$$Z_i \begin{cases} = z_i & \text{if } P_i \text{ learns no more than output of auction protocol.} \\ > z_i & \text{if } P_i \text{ learns } f_i(b_{\bar{\mathcal{C}}}) \\ < z_i' & \text{if } \exists j \neq i \text{ such that } P_j \text{ learns } f_j(b_i) \text{ for some function } f_j. \\ < z_i'' & \text{same as previous case, but } P_j \text{ also associates it with } P_i. \\ < z_i & \text{if protocol aborts.} \end{cases} \tag{4}$$

We let $z_i', z_i'' < 0$ and $z_i'' < z_i'$, emphasising that letting another party learn the identity of P_i along with its private bid information b_i is considered bigger information utility loss for P_i. The negative information utility holds even if P_i has additionally learned about $f_i(b_{\bar{\mathcal{C}}})$. This is to emphasise that parties in collusion do not prefer to lose their privacy even if it helps in learning about other parties' bids. We refer to such parties as *Privacy-sensitive* rational parties.

We present the following Lemma for an arbitrary collusion $\mathcal{C}$ formed *after* the commitments to bids (Algorithm 1.12). This does not weaken the adversarial model since if collusion were to change these commitments, the inputs to the protocol would be changed (which is out of the scope of any protocol design).

Lemma 8. *For any privacy-sensitive (Equation 2) rational PPT party P_i, let π_i be the honest strategy, and π_i' be the strategy induced on P_i while being part of collusion. Then, assuming security of bit encoding scheme,* Com *is a secure commitment scheme and NIZK proofs are sound, the strategy π_i weakly dominates the strategy π_i' for all $i \in [n]$ as per Definition 4.*

Recall that we consider rational parties participating in the SPA protocol to be privacy sensitive, regarding collusion. That is,

1. Parties join collusion only if they can make gains in their monetary/information utility.
2. Parties like to learn the information only if there is no loss of their own private information.

We will make use of these assumptions about the rational parties for the proof.

We prove this Lemma 8 in two parts. We first show that no deviation as a part of collusion increases monetary utility.

Lemma 9. *For any privacy-sensitive (Equation 4) rational PPT party P_i, let π_i be the honest strategy, and π_i' be the strategy induced on P_i while being part of collusion. Then, assuming security of bit encoding scheme,* Com *is a secure commitment scheme and NIZK proofs are sound, the strategy π_i fetches at least as much monetary utility as π_i' for all $i \in [n]$.*

Proof. Let $\mathcal{C}$ be an arbitrary collusion. We first consider the monetary utility and show that no party in a collusion can enhance its monetary utility.

The collusion $\mathcal{C}$ can only improve its monetary utility by changing the outcome of the protocol. This can be accomplished in one of the three ways. In each case, we will show that for any deviation from $\mathcal{C}$ which changes the outcome of the protocol, $\mathcal{C}$'s utility does not increase.

1. <u>Change the winner</u>: Suppose $\mathcal{C}$'s deviation changes the winner from P_i to P_j. As a result, one of the following cases occurs:

  - P_j not in $\mathcal{C}$: In this case, only P_j can have a gain in monetary utility (by Equation 4). Further, note that such a case arises only when P_i is in $\mathcal{C}$ and P_i acts differently compared to its bid commitment (either for bcbb or bcot in some rounds). Therefore P_i can not provide proofs of correct computation assuming that Π_{OT} is a secure OT protocol and NIZK proofs are sound (See Algorithms 7.17 and 7.12) and loses its security deposit, resulting in decreased monetary utility of $\mathcal{C}$.
  - P_i and P_j are both in $\mathcal{C}$: In this case, either P_i can not provide proof-of-not-winning assuming soundness of NIZK proofs (See Algorithm 7.17) or P_j can not provide proof of winner claim assuming security of bit encoding scheme (See Algorithm 7.1), hence $\mathcal{C}$'s utility decreases.
  - P_j in $\mathcal{C}$ but P_i not in $\mathcal{C}$: In this case, $b_j < b_i$. Hence P_j ends up paying higher value than its bid – resulting in reduced monetary utility for $\mathcal{C}$.
2. Change the winning price: For this, at least one of the parties in collusion say, $P_i \in \mathcal{C}$ has to deviate using bcbb, which gets detected through NIZK proofs (See Algorithm 7.17 and Lemma 7) and loses deposit. Hence the collusion loses utility
3. Attempts to implicate an honest party: One can improve the monetary gains in SPA by receiving the redistributed deposit amount of a cheating party. For this, a party $P_i \in \mathcal{C}$ may accuse another party of using different encodings in bcot and bcbb. However, as shown in Lemma 3 this attempt would result in P_i losing deposit.
4. Avoid reporting deviation: Suppose $P_i \notin \mathcal{C}$ deviates in bcot and bcbb which is detectable by $\mathcal{C}$ (See Algorithm 3 and Lemma 3). If $\mathcal{C}$ does not have the highest bidder, not reporting this would result in reduced utility. If $\mathcal{C}$ has the highest bidder in it, not reporting the deviation may result in the increased price of the auction or missing the winner discovery – reducing the overall utility of collusion.

Thus, the monetary utility of collusion only decreases with any attempt to change the outcome of the protocol. □

The only other thing a collusion can do is deviate in a way that does not affect the protocol and try to gain more information. We now show that the collusion cannot gain in information utility in this case either.

Lemma 10. *For any privacy-sensitive (Equation 4) rational PPT party P_i, let π_i be the honest strategy, and π'_i be the strategy induced on P_i while being part of collusion. Then, assuming security of bit encoding scheme,* Com *is a secure commitment scheme and NIZK proofs are sound, the strategy π_i achieves as much information utility as π'_i for all $i \in [n]$.*

Proof. By Lemma 9, a rational collusion can not increase its monetary utility (specifically, cannot change the correctness of protocol output). We first prove (in Lemma 13 below) that a deviation that is not detected and does not change the output does not affect the computation that a collusion can perform. Hence these deviations are weakly dominated by π_i, where all colluding parties stick to the protocol Π. So it is enough to consider the case when parties do not deviate, but use pool of their views of protocol $\left(\mathsf{View}_i^{\Pi}\right)_{i\in\mathcal{C}}$ (input of the computation that a collusion can do) to run arbitrary computation. We then characterize in Lemma 11 below what extra information a collusion can hope to gain (output of the computation that a collusion can get), when all parties have individual utilities as mentioned in Equation 4. Further, in order, for the collusion to increase its information utility, both the Highest bidder (HB) and the Second highest bidder (SHB) should be part of it (Lemma 12 below).

Now that we have established that a collusion that does not include both SHB and HB is not rational, and characterized the only computation this collusion can perform (OR of bits of parties outside collusion, by Lemma 11), we argue that it is irrational for SHB to be a part of this collusion if anyone other than SHB gets the output (characterizing who can get the output). If anyone apart from SHB gets the output (OR of non-colluding parties), then they can conclude that SHB is a part of collusion and hence decreases the information utility of SHB (Equation 4). To summarize, a set of colluding parties can run an MPC outside the protocol with their views as input and obtain OR of bits of non colluding parties as output, where this output is given only to SHB. With the above observations, the following cases are possible:

- $\mathcal{C}$ formed after protocol: In this case, no party other than SHB would collude as they cannot increase their information utility. Hence, no party other than SHB would collude.
- $\mathcal{C}$ formed during protocol: Let P' be a party in $\mathcal{C}$ that does not end up as HB or SHB. Since P' does not learn anything, P' has no chance to increase its own utility. But Suppose parties, other than P',

in $\mathcal{C}$ form another collusion, C', to carry out the same exercise. Suppose SHB, as part of C', learns B'. The following cases are possible:

- $B' > B$: In this case, SHB would be able to learn that the specific party P' has the largest bits outside C'. But if P' never participated in any collusion to begin with, SHB only learns bits of P' but would not be able to associate it to P'.
- $B' = B$: In this case, SHB learns that bits used by P' are strictly lower than B. If P' did not participate in any collusion to begin with, SHB can not learn this information.

Since this argument holds for any party that is not HB or SHB, it is irrational for any privacy sensitive party to participate in any collusion. Here we note that if the party ends up to be neither HB nor SHB, then there is a strict reduction in its information utility due to the previous case. Since a party follows a certain strategy only if that does not reduce its utility, again, such a collusion is irrational.

To summarize, observe that the collusion can be formed either during the protocol run or after the protocol run. During the protocol run, since parties lose their information utility in case they do not end up being SHB, they desist from colluding. After the protocol run, no party other than SHB has any chance to gain information utility by forming collusion and hence no collusion is formed. Thus it follows that $\forall i \in \mathcal{C},\ Z_i \leq z_i$.

Hence, forming collusion for gains in information utility is weakly dominated by acting individually. □

Lemma 11. *Suppose every party acts as per Π, i.e., follows the SPA protocol, then as a result of utilities defined in Equation 4, and assuming security of bit encoding scheme,* Com *is a secure commitment scheme and NIZK proofs are zero knowledge, the only information a collusion can learn is OR of bits contributed by non-colluding parties.*

Proof. Because of utilities defined in Equation 4, the output of any joint computation that a collusion performs on their views should only contain information about parties outside the collusion (as it is irrational for the colluding party to reveal information about itself). The only information to be learned about parties outside collusion is to learn about their bids. Now, note that the hiding property of the commitment scheme Com ensures that the bit commitments written to BB do not leak any information to PPT parties (and their collusion). Also, by zero knowledge property of the proofs, the NIZK proofs also do not leak anything about values committed. Hence, we consider the collusion $\mathcal{C}$ to use the encoded bits written to BB for information gain. From Lemma 15 we have that $\mathcal{C}$ can learn no more than the bits contributed by the highest bidder outside $\mathcal{C}$, without learning its identity. □

Lemma 12. *Suppose every party acts as per Π, i.e., follows the SPA protocol, then assuming security of bit encoding scheme,* Com *is a secure commitment scheme and NIZK proofs are zero knowledge, a collusion of parties can not learn anything unless both HB and SHB are part of the collusion.*

Proof. Recall that HB contributes exactly the same bits as that of the second highest bid. By the previous lemma, any collusion that does not have both SHB and HB in it can learn only the SH bid value. □

Lemma 13. *Suppose no party in the collusion deviates such that their deviation is detected or changes the output of the protocol, then the collusion can not learn anything more about a party outside the collusion compared to the case where every party in the collusion follows the protocol.*

Proof. A careful scrutiny of the protocol shows that the following are the only deviations that neither get detected by a party outside collusion nor change the output of the protocol:

– Deviations in OT communication between P_i and P_j such that both $P_i, P_j \in \mathcal{C}$: Such a deviation does not affect the view of a non-colluding party and hence does not change messages sent by that party.

– Deviation in sampling randomness: Other parties' messages are unaffected by such a deviation from any party in the collusion (their message is only a function of d_{ij} computation that they locally perform, and any deviation to change this computation gets detected)

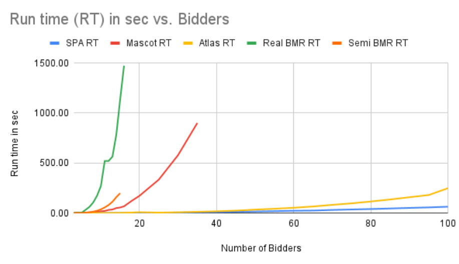
The above points show that the encodings received (from non colluding parties) by every party in the collusion remain unaffected. Since the encodings of non-colluding parties remain the same as honest, by Lemma 15, the collusion can learn no more than OR of bits of non-colluding parties (as in Lemma 11). □

7 Experimental results

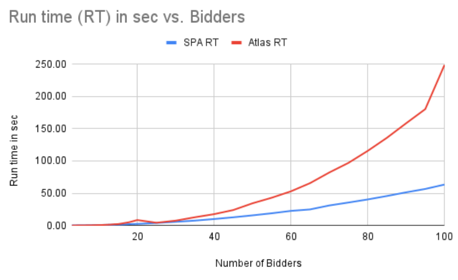
Our protocol was implemented in C++ and executed on a single machine using Intel core i7 processor with 32GB RAM, 2.9 GHz, running Ubuntu 22.04 operating system. Each bidder is represented by a separate process. We have used the elliptic curve secp256k1 as the underlying Group from OpenSSL. NUMS implementation is used for computing the group generators. We have implemented the NIZK proofs required for our protocol (for relation 8 in Appendix B) using the construction from [CS97,DGP22]. We instantiate our Oblivious Transfer with the construction in Figure 13 of [CSW20]. We make our implementation available at `https://github.com/girishabs/spa-code`.
In the following, we compare our protocol's run time and communication cost with the protocols from MP-SPDZ suite [Kel20] in Table 2. For each protocol, the measurements are taken for 15 bidders with each bid $100 \leq b < 1000$. We observe that our protocol runs 1.7X faster than the semi-honest protocol *Atlas*.

Protocol	**Threat model**	**Run time (in sec)**	**Communication Complexity (in MB)**
Our Protocol	Rational	1.26	0.779
Atlas [GLO+21]	Semi-honest	2.11	6.09
Mascot [KOS16]	Malicious	55.65	4074.5
Semi-BMR [LPSY19]	Semi-honest	198.77	18462
Real-BMR [LPSY19]	Malicious	1137.39	109905

Table 2: Comparison of the efficiency of protocols.



(a) Protocol Run times vs number of bidders



(b) Protocol Run times for SPA and Atlas compared

Figure 2b shows the plot of run times of various protocols for different numbers of bidders. For BMR protocols, the maximum number of bidders we could consider while running on our machine was 15, and for Mascot, it was 35.

8 Conclusion

We construct a protocol for *Second Price Auction* that is provably secure in the rational setting and concretely efficient. We utilize the notion of *Privacy preserving computational dominant strategy equilibrium* to establish rational security. Our work leaves open several interesting questions about extending the protocol to other flavors of auctions, such as multi-unit auctions, analyzing other adversarial models like adaptive strategies and designing broadcast protocols with rational parties.

Acknowledgments

We thank the anonymous reviewers of CCS 2024 for their careful reading of the paper and helpful feedback. This research is supported, in part, by SERB Grant SP/SERB-23-0328. The research of the first author is supported by Rising Star Award, Intel Corporation and Google India Faculty Research Award.

References

ADGH06. Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In Eric Ruppert and Dahlia Malkhi, editors, *25th ACM PODC*, pages 53–62. ACM, July 2006.

BGM$^+$18. Christian Badertscher, Juan A. Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But why does it work? A rational protocol design treatment of bitcoin. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 34–65. Springer, Heidelberg, April / May 2018.

BHSR19. Samiran Bag, Feng Hao, Siamak F Shahandashti, and Indranil Ghosh Ray. Seal: Sealed-bid auction without auctioneers. *IEEE Transactions on Information Forensics and Security*, 15:2042–2052, 2019.

Bra01. Felix Brandt. Cryptographic protocols for secure second-price auctions. In *International Workshop on Cooperative Information Agents*, pages 154–165. Springer, 2001.

BYK21. Osman Biçer, Burcu Yildiz, and Alptekin Küpçü. m-stability: Threshold security meets transferable utility. In *Proceedings of the 2021 on Cloud Computing Security Workshop*, pages 73–82, 2021.

CH13. Boaz Catane and Amir Herzberg. Secure second price auctions with a rational auctioneer. In *2013 International Conference on Security and Cryptography (SECRYPT)*, pages 1–12. IEEE, 2013.

CS97. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

CSW20. Ran Canetti, Pratik Sarkar, and Xiao Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 277–308. Springer, Heidelberg, December 2020.

DGP22. Bernardo David, Lorenzo Gentile, and Mohsen Pourpouneh. Fast: fair auctions via secret transactions. In *International Conference on Applied Cryptography and Network Security*, pages 727–747. Springer, 2022.

DHR00. Yevgeniy Dodis, Shai Halevi, and Tal Rabin. A cryptographic solution to a game theoretic problem. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 112–130. Springer, Heidelberg, August 2000.

DM16. Alan Deckelbaum and Silvio Micali. Collusion, efficiency, and dominant strategies. 2016.

GK06. S. Dov Gordon and Jonathan Katz. Rational secret sharing, revisited. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 229–241. Springer, Heidelberg, September 2006.

GKM$^+$13. Juan A. Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *54th FOCS*, pages 648–657. IEEE Computer Society Press, October 2013.

GKS22. Chaya Ganesh, Bhavana Kanukurthi, and Girisha Shankar. Secure auctions in the presence of rational adversaries. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1173–1186, 2022.

GKTZ12. Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP 2012, Part II*, volume 7392 of *LNCS*, pages 561–572. Springer, Heidelberg, July 2012.

GKTZ15. Juan A. Garay, Jonathan Katz, Björn Tackmann, and Vassilis Zikas. How fair is your protocol? A utility-based approach to protocol optimality. In Chryssis Georgiou and Paul G. Spirakis, editors, *34th ACM PODC*, pages 281–290. ACM, July 2015.

GLO$^+$21. Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. ATLAS: Efficient and scalable MPC in the honest majority setting. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 244–274, Virtual Event, August 2021. Springer, Heidelberg.

Hal08. Joseph Y. Halpern. Beyond nash equilibrium: solution concepts for the 21st century. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *27th ACM PODC*, pages 1–10. ACM, August 2008.

HM21. Po-Chu Hsu and Atsuko Miyaji. Bidder scalable m+ 1st-price auction with public verifiability. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 34–42. IEEE, 2021.

HP10. Joseph Y. Halpern and Rafael Pass. Game theory with costly computation: Formulation and application to protocol security. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 120–142. Tsinghua University Press, January 2010.

HT04. Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: Extended abstract. In László Babai, editor, *36th ACM STOC*, pages 623–632. ACM Press, June 2004.

HZ06. Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *International Workshop on Security Protocols*, pages 202–211. Springer, 2006.

IML05. Sergei Izmalkov, Silvio Micali, and Matt Lepinski. Rational secure computation and ideal mechanism design. In *46th FOCS*, pages 585–595. IEEE Computer Society Press, October 2005.

Kat08. Jonathan Katz. Bridging game theory and cryptography: Recent results and future directions (invited talk). In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 251–272. Springer, Heidelberg, March 2008.

Kel20. Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1575–1590, 2020.

KN08. Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 320–339. Springer, Heidelberg, March 2008.

KO02. Kaoru Kurosawa and Wakaha Ogata. Bit-slice auction circuit. In *Proceedings of the 7th European Symposium on Research in Computer Security*, pages 24–38, 2002.

KOS16. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016.

Kri09. Vijay Krishna. Auction theory. 2009.

LAN02. Helger Lipmaa, N Asokan, and Valtteri Niemi. Secure vickrey auctions without threshold trust. In *Proceedings of the 6th international conference on Financial cryptography*, pages 87–101, 2002.

LPSY19. Yehuda Lindell, Benny Pinkas, Nigel P Smart, and Avishay Yanai. Efficient constant-round multi-party computation combining bmr and spdz. *Journal of Cryptology*, 32(3):1026–1069, 2019.

Mic14. Silvio Micali. Rational and resilient protocols. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, page 1. ACM, July 2014.

MNT09. Peter Bro Miltersen, Jesper Buus Nielsen, and Nikos Triandopoulos. Privacy-enhancing auctions using rational cryptography. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 541–558. Springer, Heidelberg, August 2009.

MR14. Silvio Micali and Michael O. Rabin. Cryptography miracles, secure auctions, matching problem verification. *Commun. ACM*, 57(2):85–93, 2014.

Nar14. Yadati Narahari. *Game theory and mechanism design*, volume 4. World Scientific, 2014.

NPS99. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999.

NS14. Mehrdad Nojoumian and Douglas R Stinson. Efficient sealed-bid auction protocols using verifiable secret sharing. In *International Conference on Information Security Practice and Experience*, pages 302–317. Springer, 2014.

OM03. Kazumasa Omote and Atsuko Miyaji. A second-price sealed-bid auction with the discriminant of the p0-th root. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 57–71. Springer, Heidelberg, March 2003.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

Sch22. Nikolaj Ignatieff Schwartzbach. Payment schemes from limited information with applications in distributed computing. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 129–149, 2022.

A Anonymous Bidding Protocol

We use the *Anonymous Bidding Protocol* (ABP) to compute the highest bid, which is a variation of the Anonymous Veto Protocol (AVP) first described in [HZ06] and later used in [BHSR19,DGP22,GKS22].

ABP runs for l rounds – where l is the number of bits in the binary representation of the bids. The protocol proceeds as below:

– Let P_i's bid $b_i = (b_{i1}||\dots||b_{il})$.

– P_i participates in jth round of ABP by contributing b_{ij}. This continues until P_i keeps contributing a bit equal to the computed output for the round. If in a round, there is other bidder who contributes a 1 while the P_i has 0, P_i drops out of the race at the end of that round.

– A party who has dropped out of the race, continues to participate in the protocol but only contributes a 0 for all remaining rounds.

– Any round $1 \leq j \leq l$ which has at least one bidder bidding a 1 bit is considered as the *decider round.*

– Thus, a bidder P_i uses the bid d_{ij} during round j as

$$d_{ij} = \begin{cases} 0, & \text{if } P_i \text{ is not in race} \\ & \text{ or } P_i \text{ bid 0 in any of previous decider rounds.} \\ b_{ij}, & \text{if } P_i \text{ is in the race} \\ & \text{ or } P_i \text{ bid 1 in all previous decider rounds.} \end{cases}$$

– Logical OR of all individual bids used in jth round is evaluated to be the jth computed bit. i.e.,

$$b_{sj} = \bigvee_{i=1}^{n} d_{ij}$$

– Protocol's output bid b_s is computed as $b_s = b_{s1}||\cdots||b_{sl}$

Security of encoding: In order to ensure the privacy of the bits used during computation, the contribution bits are encoded such that no PPT party can distinguish between the encoding of 0 and the encoding of 1. There can be several possible encodings for representing the bits. For example, parties can use a secret share of 0 to represent 0-bit encoding and a secret share of any non-zero number to represent 1-bit encoding. The reconstructed value would represent the encoding of the computed bit. Notice that this scheme does indeed satisfy the requirements of ABP computation described above.

However, we use the encoding scheme adopted from [DGP22]. For this, we make use of a group $\mathbb{G}$ of prime order q where DDH assumption holds. Let $g \in \mathbb{G}$ be a publicly known generator. The encoding and corresponding computation on the coded bits are performed as follows:

1. Each bidder P_i, $i \in [n]$ allocates private keys $x_{ij}, r_{ij} \xleftarrow{\$} \mathbb{Z}_q$, $i \in [n], j \in [l]$. Public keys $X_{ij} = g^{x_{ij}}$ are published to the bulletin board.
2. Once public keys from all bidders are available, each bidder computes:
$$Y_{ij} = \frac{\prod_{k=1}^{i-1} X_{kj}}{\prod_{k=i+1}^{n} X_{kj}}$$
3. Each *contributed* bit d_{ij} is encoded as :
$$B_{ij} = \begin{cases} \text{0-bit code :} & Y_{ij}^{x_{ij}} \quad \text{if} \quad d_{ij} = 0 \\ \text{1-bit code :} & g^{r_{ij}} \quad \text{if} \quad d_{ij} = 1 \end{cases}$$
4. The jth computed bit is evaluated as the logical-OR of individual bidding bits b_{ij} for the jth position: $b_{sj} = \bigvee_{i=1}^{n} b_{ij}$. This is computed using the encoded bits as follows:
$$b_{sj} = \begin{cases} 0, & \text{if} \quad \prod_{i=1}^{n} B_{ij} = 1 \\ 1, & \text{if} \quad \prod_{i=1}^{n} B_{ij} \neq 1 \end{cases} \tag{5}$$

Assuming DDH assumption holding in $\mathbb{G}$, for all PPT parties, the encoding of 0 and 1 are indistinguishable.

Lemma 14. *Assuming DDH assumption holds in $\mathbb{G}$, for all pair of PPT parties $P_k, P_i, k \neq i$, following two distributions are indistinguishable:*

1. $(g, (X_1, \ldots, X_n), Y_i, Y_i^{x_i})$
2. $(g, (X_1, \ldots, X_n), Y_i, g_i^{r_i})$

where $g \in \mathbb{G}$ is the generator of group $\mathbb{G}$, $X_k = g^{x_k}, x_k \in \mathbb{Z}_q, \forall k \in [n], r_i \in \mathbb{Z}_q$ and $Y_i = \dfrac{\prod_{k=1}^{i-1} X_k}{\prod_{k=i+1}^{n} X_k}$. Moreover, $Y_i^{x_i}$ and $g_i^{r_i}$ are 0 and 1 encodings of party P_i.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ who can distinguish between the aforementioned distributions of certain party P_i. We will now construct another adversary $\mathcal{B}$ who can break DDH assumption. The reduction is as follows.

– DDH challenger chooses $b \xleftarrow{\$} \{0,1\}$ and $u, v, w \xleftarrow{\$} \mathbb{Z}_q$. Sets $U = g^u, V = g^v$. If $b = 0$, sets $W = g^{uv}$ and if $b = 1$, sets $W = g^w$.

– $\mathcal{B}$ playing the DDH attack game receives the challenge (g, U, V, W).

– For P_i whose encoded bits are to be distinguished, $\mathcal{B}$ sets $X_i = g^u$.
  $\mathcal{B}$ chooses $X_k \xleftarrow{\$} \mathbb{G}, \forall k \neq i, k \neq 1$. Sets $X_1 = V \cdot \dfrac{\prod_{k=i+1}^{n} X_k}{\prod_{k=2}^{i-1} X_k}$.

– $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine and sends across
  $(g, (X_1, \ldots, U, \ldots, X_n), V, W)$ to $\mathcal{A}$.

– $\mathcal{A}$, being the $0,1$-encoding distinguisher returns its guess b' to $\mathcal{B}$, who returns same bit as its guess to DDH challenger.

Notice that since $\mathcal{A}$ is assumed to be PPT distinguisher, $\mathcal{B}$ also runs in polynomial time. Consider the two cases:

$b = 0$: In this case, view of $\mathcal{A}$ is $(g, (X_1, \ldots, X_n), Y_i, Y_i^{x_i})$ which corresponds to distribution with P_i's 0-bit encoding where $Y_i = g^v$, $v = \left(\sum_{k=1}^{i-1} x_k - \sum_{i+1}^{n} x_k\right)$, $X_k = g^{x_k}, \forall k \neq i$, $X_i = g^u$ and $x_i = u$.

$b = 1$: In this case, view of $\mathcal{A}$ is $(g, (X_1, \ldots, X_n), Y_i, g^{r_i})$ which corresponds to distribution with P_i's 1-bit encoding, where $Y_i = g^v$, $v = \left(\sum_{k=1}^{i-1} x_k - \sum_{i+1}^{n} x_k\right)$, $X_k = g^{x_k}, \forall k \neq i$, $X_i = g^u$, $x_i = u$ and $r_i = w$.

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins the game}] \\
&= \Pr[b' = 1 | b = 1]\Pr[b = 1] + \Pr[b' = 0 | b = 0]\Pr[b = 0] \\
&= \frac{1}{2}\left(\Pr[b' = 1 | b = 1] + \Pr[b' = 0 | b = 0]\right) \\
&= \frac{1}{2}\left(\Pr[b' = 1 | b = 1] + 1 - \Pr[b' = 1 | b = 0]\right) \\
&= \frac{1}{2} + \frac{1}{2}\left(\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]\right) \\
&= \frac{1}{2} + \frac{1}{2}\Pr[\text{Distinguishing advantage of DDH adversary } \mathcal{B}]
\end{aligned}$$

However, since DDH assumption holds in $\mathbb{G}$,

$$\Pr[\text{Distinguishing advantage of DDH adversary } \mathcal{B}] \leq \mathsf{negl}$$

Hence we have $\Pr[\mathcal{A} \text{ wins the game}] \leq \dfrac{1}{2} + \mathsf{negl}$ □

In addition, this encoding also ensures that any collusion $\mathcal{C}$ of parties does not learn anything more than the highest bid among the remaining parties $\bar{\mathcal{C}}$.

Lemma 15. *Let $P_1, \ldots, P_n$ be a set of PPT parties. Let each P_i sample $b_i \xleftarrow{\$} \{0,1\}$, encode B_i as per the bit encoding scheme mentioned above and write the same to BB. Let $\mathcal{C}$ be the set of colluding parties. Assuming security of bit encoding scheme, the information learnt by $\mathcal{C}$ is $\bigvee_{k \in \bar{\mathcal{C}}} b_k$ where b_k are the bits sampled by $P_k \in \bar{\mathcal{C}}$.*

Proof. Recall that each party P_i writes the encoded bit B_i to the BB. This enables the parties to evaluate the OR of the contributed bits as in Equation 5. Also, observe that this is the only computation that is defined on these encoded bits. Hence, the possible leakage is when the parties in $\mathcal{C}$ use their private inputs along with the encoded bits written on BB.

Parties in collusion have a choice to either use encoding of 1-bit or 0-bit for such a computation. However, since the computation in Equation 5 is for the OR of the computed bits, using 1-bit encoding masks the inputs of parties in $\bar{\mathcal{C}}$. Since parties in collusion $\mathcal{C}$ are interested in learning about the bits in bids of parties in $\bar{\mathcal{C}}$, they pool their 0-encoding and evaluate the product $W_{\bar{\mathcal{C}}}$ as follows:

$$W_{\bar{\mathcal{C}}} = \prod_{k \in \mathcal{C}} [0]_k \cdot \prod_{k \in \bar{\mathcal{C}}} B_k \tag{6}$$

Two cases to be considered (as per the Equation 5):

In the first case $W_{\bar{\mathcal{C}}} = 1$, since each party in $\mathcal{C}$ has contributed the 0 encoding, it must be the case that every party in $\bar{\mathcal{C}}$ has also contributed a 0. Thus in this case, the bit learnt by $\mathcal{C}$ is indeed $\bigvee_{k \in \bar{\mathcal{C}}} b_k$.

In the second case $W_{\bar{\mathcal{C}}} \neq 1$, at least one party in $\bar{\mathcal{C}}$ has used a 1-bit. In the following, we will argue that $\mathcal{C}$ cannot learn either who from $\bar{\mathcal{C}}$ has contributed 1 or how many parties in $\bar{\mathcal{C}}$ have contributed 1. For this, we will show that the distribution where some parties in $\bar{\mathcal{C}}$ (say q of them) contribute a 1-bit encoding is indistinguishable from the distribution where at most one of the parties in $\bar{\mathcal{C}}$ contribute 1.

Let $R = \left\{ B_i \mid P_i \in \bar{\mathcal{C}}, B_i = g^{r_i} \right\}$, i.e., the set of parties in $\bar{\mathcal{C}}$ who have contributed a 1 during round j. Let $|R| = q$. View of the collusion $\mathcal{C}$ (excluding elements unrelated to the encoding of bits),

$$\mathsf{View}_{\mathcal{C}} = \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (B_i)_{i \in R} \right) \tag{7}$$

We consider the following hybrids:
H_0 : $\mathsf{View}_{\mathcal{C}_0} = \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (B_{i_1}, \dots, B_{i_q}) \right)$
$= \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (g^{r_{i_1}}, \dots, g^{r_{i_q}}) \right)$

H_1 : is same as H_0 except that P_{i_1}'s 1-bit encoding is replaced with 0-bit encoding.
$\mathsf{View}_{\mathcal{C}_1} = \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (Y_{i_1}^{x_{i_1}}, \dots, g^{r_{i_q}}) \right)$
H_1 is indistinguishable from H_0 from Lemma 14.

H_2 : is same as H_1 except that P_{i_2}'s 1-bit encoding is replaced with 0-bit encoding.
$\mathsf{View}_{\mathcal{C}_2} = \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (Y_{i_1}^{x_{i_1}}, Y_{i_2}^{x_{i_2}}, \dots, g^{r_{i_q}}) \right)$
H_2 is indistinguishable from H_1 from Lemma 14.

$\vdots$ H_m : is same as H_{m-1} except that P_{i_m}'s 1-bit encoding is replaced with 0-bit encoding.
$\mathsf{View}_{\mathcal{C}_m} = \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (Y_{i_1}^{x_{i_1}}, Y_{i_2 j}^{x_{i_2}}, \dots, Y_{i_m}^{x_{i_m}}, \dots, g^{r_{i_q}}) \right)$
H_m is indistinguishable from H_{m-1} from Lemma 14.

$\vdots$ H_{q-1} : is same as H_{q-1} except that $P_{i_{q-1}}$'s 1-bit encoding is replaced with 0-bit encoding.
$\mathsf{View}_{\mathcal{C}_{q-1}} = \left((g^{x_i})_{i \in [n]}, (B_i)_{i \in \mathcal{C}}, (B_i)_{i \in \bar{\mathcal{C}} \setminus R}, (Y_{i_1}^{x_{i_1}}, \dots, Y_{i_m}^{x_{i_m}}, \dots, Y_{i_{q-1}}^{x_{i_{q-1}}}, g^{r_{i_q}}) \right)$
H_{q-1} is indistinguishable from H_{q-2} from Lemma 14.

Notice that H_0 has all members of R contributing 1-bit encoding whereas H_{q-1} has all members except $P_{i_q} \in R$ contributing 0-bit encoding. By transitivity, $H_0 \approx_c H_{q-1}$. Moreover, the computed value $W_{\bar{\mathcal{C}}} \neq 1$ for each of the hybrids. Thus it follows that $\mathcal{C}$ cannot learn the number of parties in $\bar{\mathcal{C}}$ who have contributed 1. Moreover, parties $(P_{i_1}, \dots, P_{i_{q-1}})$ change from using 1-bit encoding in H_0 to 0-encoding in H_{q-1}. Yet the two distributions remain indistinguishable. Thus $\mathcal{C}$ cannot learn the information as to who among $\bar{\mathcal{C}}$ would have contributed 1.

The only bit learnt by $\mathcal{C}$ is $\bigvee_{k \in \bar{\mathcal{C}}} b_k$. Moreover, as shown in the correctness Theorem 1, the computed value is the highest bit among the contributed bits. Since the parties in $\mathcal{C}$ have contributed a 0 bit encoding, it follows that the computed values by $\mathcal{C}$ are indeed the bits contributed by the highest bidder in $\bar{\mathcal{C}}$. □

B NIZK Proofs

This section describes the NIZK proof for losing parties. This proof is used to prove that a party has correctly computed the bit code written onto BB during every decider round j. For the construction of these NIZK proofs, we assume that *Pedersen Commitment scheme* [Ped92] is used in the protocol. Each losing bidder P_i needs to prove the following for jth iteration:

- Proof of commitment: $c_{ij} = \mathsf{Com}(b_{ij}, a_{ij}) = g^{b_{ij}} h^{a_{ij}}$
- Proof of knowledge of secret keys used for computation: $B_{ij} = Y_{ij}^{x_{ij}}$ for the 0-bit code and $B_{ij} = g^{r_{ij}}$ for the 1-bit code.
- Proof of knowledge of secret key used for computation during the previous decider round $\bar{j}$, $B_{i\bar{j}} = Y_{i\bar{j}}^{x_{i\bar{j}}}$ for the 0-bit code or $B_{i\bar{j}} = g^{r_{i\bar{j}}}$ for the 1-bit code. Along with this, knowledge of actual secret keys used for computation during iterations $\bar{j}, j$ also need to be proved.

Thus, the relation that is established by this NIZK is:

$$\mathcal{R} = \left\{ \begin{matrix} b_{ij}, a_{ij}, \\ x_{ij}, r_{i\bar{j}}, \\ r_{ij}, x_{i\bar{j}} \end{matrix} \middle| \begin{matrix} \left(\frac{c_{ij}}{g} = h^{a_{ij}} \wedge B_{i\bar{j}} = g^{r_{i\bar{j}}} \wedge B_{ij} = g^{r_{ij}}\right) \bigvee \\ \left(c_{ij} = h^{a_{ij}} \wedge B_{ij} = Y_{ij}^{x_{ij}} \wedge X_{ij} = g^{x_{ij}}\right) \bigvee \\ \left(\begin{matrix}\frac{c_{ij}}{g} = h^{a_{ij}} \wedge B_{i\bar{j}} = Y_{i\bar{j}}^{x_{i\bar{j}}} \wedge X_{i\bar{j}} = g^{x_{i\bar{j}}} \wedge \\ B_{ij} = Y_{ij}^{x_{ij}} \wedge X_{ij} = g^{x_{ij}}\end{matrix}\right) \end{matrix} \right\} \tag{8}$$

We use the approach in [CS97,DGP22] for constructing NIZK for the above relation. This is accomplished by representing the relation in terms of the following clauses:

$$\begin{aligned} F_1 =& DL(h, c_{ij}) \otimes [DL(Y_{ij}, B_{ij}) \cap DL(g, X_{ij})] \\ F_2 =& DL(h, c_{ij}/g) \otimes DL(g, B_{i\bar{j}}) \otimes DL(g, B_{ij}) \\ F_3 =& DL(h, c_{ij}/g) \otimes \left[DL(Y_{i\bar{j}}) \cap DL(g, X_{i\bar{j}})\right] \otimes \\ & [DL(Y_{ij}, B_{ij}) \cap DL(g, X_{ij})] \end{aligned}$$

Thus, the proofs need to be constructed for $F = F_1 \cup F_2 \cup F_3$.

To prove the knowledge of either F_1 or F_2 or F_3, assuming that F_k is known, party P_i proceeds as follows:

1. Choose $v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7 \xleftarrow{\$} \mathbb{Z}_q$.
2. Choose (w_0, w_1, w_2) with $w_k = 0$ and $w_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \neq k$.
3. Compute the commitment tokens:
   - $t_0 = c_{ij}^{w_0} h^{v_0}, \quad t_1 = B_{ij}^{w_0} Y_{ij}^{v_1}, \quad t_2 = X_{ij}^{w_0} g^{v_1}$
   - $t_3 = \left(\frac{c_{ij}}{g}\right)^{w_1} h^{v_2}, \quad t_4 = B_{i\bar{j}}^{w_1} g^{v_3}, \quad t_5 = B_{ij}^{w_1} g^{v_4},$
   - $t_6 = \left(\frac{c_{ij}}{g}\right)^{w_2} h^{v_5}, \quad t_7 = B_{i\bar{j}}^{w_2} Y_{i\bar{j}}^{v_6},$
   - $t_8 = X_{i\bar{j}}^{w_2} g^{v_6}, \quad t_9 = B_{ij}^{w_2} Y_{ij}^{v_7}, \quad t_{10} = X_{ij}^{w_2} g^{v_7}$
4. Compute the random challenge using the public hash function as:
$$H = \mathcal{H}\left(\begin{matrix} h, c_{ij}, Y_{ij}, B_{ij}, g, X_{ij}, \frac{c_{ij}}{g}, B_{i\bar{j}}, Y_{i\bar{j}}, X_{i\bar{j}}, \\ t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \end{matrix}\right) \mod q$$
5. Compute $(\gamma_0, \gamma_1, \gamma_2)$ as:
$$\gamma_i = \begin{cases} H - (w_0 + w_1 + w_2) \mod q, & \text{if } i = k \\ w_i, & \text{otherwise} \end{cases}$$
6. Set:
$$(u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_7) = \begin{cases} (a_{ij}, x_{ij}, 0, 0, 0, 0, 0, 0), & k = 1 \\ (0, 0, a_{ij}, r_{i\bar{j}}, r_{ij}, 0, 0, 0), & k = 2 \\ (0, 0, 0, 0, 0, a_{ij}, x_{i\bar{j}}, x_{ij}), & k = 3 \end{cases}$$
7. Compute responses $R = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10})$ as (all operations are modulo q):

- $s_0 = v_0 - \gamma_k u_0 \quad s_1 = s_2 = v_1 - \gamma_k u_1$
- $s_3 = v_2 - \gamma_k u_2 \quad s_4 = v_3 - \gamma_k u_3 \quad s_5 = v_4 - \gamma_k u_4$
- $s_6 = v_5 - \gamma_k u_5 \quad s_7 = s_8 = v_6 - \gamma_k u_6 \quad s_9 = s_{10} = v_7 - \gamma_k u_7$

8. Publish the proof as:
   $\pi = (\gamma_0, \gamma_1, \gamma_2, s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10})$.
9. Proof validity can be checked by reconstructing the commitments:
   - $t'_0 = c_{ij}^{\gamma_0} h^{s_0}, \quad t'_1 = B_{ij}^{\gamma_0} Y_{ij}^{s_1}, \quad t'_2 = X_{ij}^{\gamma_0} g^{s_2}$
   - $t'_3 = (c_{ij}/g)^{\gamma_1} h^{s_3}, \quad t'_4 = B_{i\bar{j}}^{\gamma_1} g^{s_4}, \quad t'_5 = B_{ij}^{\gamma_1} g^{s_5}$
   - $t'_6 = (c_{ij}/g)^{\gamma_2} h^{s_6}, \quad t'_7 = B_{i\bar{j}}^{\gamma_2} Y_{i\bar{j}}^{s_7}$
   - $t'_8 = X_{i\bar{j}}^{\gamma_2} g^{s_8}, \quad t'_9 = B_{ij}^{\gamma_2} Y_{ij}^{s_9}, \quad t'_{10} = X_{ij}^{\gamma_2} g^{s_{10}}$
10. Evaluate:
$$H' = \mathcal{H}\left(\substack{h, c_{ij}, Y_{ij}, B_{ij}, g, X_{ij}, \frac{c_{ij}}{g}, B_{i\bar{j}}, Y_{i\bar{j}}, X_{i\bar{j}}, \\ t'_0, t'_1, t'_2, t'_3, t'_4, t'_5, t'_6, t'_7, t'_8, t'_9, t'_{10}}\right) \mod q$$
11. Then check for the following condition:
$$\gamma_0 + \gamma_1 + \gamma_2 \stackrel{?}{=} H'$$
12. Accept if the check passes.