Secure Auctions in the Presence of Rational Adversaries

Chaya Ganesh, Bhavana Kanukurthi, and Girisha Shankar

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

Abstract. Sealed bid auctions are used to allocate a resource among a set of interested parties. Traditionally, auctions need the presence of a trusted auctioneer to whom the bidders provide their private bid values. Existence of such a trusted party is not an assumption easily realized in practice. Generic secure computation protocols can be used to remove a trusted party. However, generic techniques result in inefficient protocols, and typically do not provide fairness – that is, a corrupt party can learn the output and abort the protocol thereby preventing other parties from learning the output.

At CRYPTO 2009, Miltersen, Nielsen and Triandopoulos [MNT09], introduced the problem of building auctions that are secure against rational bidders. Such parties are modeled as self-interested agents who care more about maximizing their utility than about learning information about bids of other agents. To realize this, they put forth a novel notion of *information utility* and introduce a game-theoretic framework that helps analyse protocols while taking into account both *information utility* as well as *monetary utility*. Unfortunately, their construction makes use a of generic MPC protocol and, consequently, the authors do not analyze the concrete efficiency of their protocol.

In this work, we construct the first concretely efficient and provably secure protocol for First Price Auctions in the *rational* setting. Our protocol guarantees privacy and fairness. Inspired by [MNT09], we put forth a solution concept that we call *Privacy Enhanced Computational Weakly Dominant Strategy Equilibrium* that captures parties' privacy and monetary concerns in the game theoretic context, and show that our protocol realizes this. We believe this notion to be of independent interest.

Our protocol is crafted specifically for the use case of auctions, is simple, using off-the-shelf cryptographic components. Executing our auction protocol on commodity hardware with 10 bidders, with bids of length 10, our protocol runs to completion in $0.141s$ and has total communication of 30KB.

Keywords: Auctions, rational adversaries

Table of Contents

1 Introduction . . . 3
1.1 Technical Overview . . . 4
1.2 Related Work . . . 7
1.3 Comparison with other approaches . . . 7
2 Preliminaries . . . 8
2.1 Equilibrium notions . . . 8
2.2 Building Blocks . . . 9
3 Auction Protocol . . . 10
3.1 Security Model . . . 10
3.2 Anonymous Bidding Protocol . . . 10
3.3 Bit Encoding Scheme . . . 11
3.4 Our Protocol . . . 11
4 Correctness of Protocol . . . 14
5 Security against Rational adversaries . . . 15
5.1 Existence of Weakly Dominant Strategy Equilibrium (weak DSE) . . . 16
5.2 Strategies for Rational Parties . . . 16
5.3 Information Utility . . . 19
5.4 Privacy Enhanced weak DSE . . . 21
6 Experimental Results . . . 24
7 Conclusion . . . 25
A Oblivious Transfer Instantiation. . . . 28
B Contract Functionality . . . 28

1 Introduction

From Psychology, to Economics, to Computer Science, the wide perspectives from which auctions have been studied is a testimony to their relevance in society. Auctions have been in vogue since time immemorial and [Kri09] refers to auctions as early as 500 BC. In fact, the Roman Empire was auctioned in the year 193AD! While the modern age offers far less dramatic use-cases of auctions, the ubiquity and the monetary stakes of the use-cases make up for it. Governments accrue huge revenues through spectrum auctions. Sotheby's auctions for the art works have been legendary in terms of works that have gone under their hammer. Search engines such as Google use auctions to determine which advertisements show up for a certain search and in what order. Broadcasting rights for sporting events are distributed through auctions. Sports franchisees use auctions for player-selection. It would not be too far from the truth to say that what can be auctioned is limited only by one's imagination.

An auction consists of a set of parties (also known as bidders) who are bidding for a particular object or resource. Such parties submit their bids i.e., the amount at which the parties are willing to buy the auctioned item. There are mainly two types of auctions: sealed bid and open bid auctions. As the name itself suggests, in the case of open bid auctions, the individual bids are not private. In sealed bid auctions, we can have either *First Price Auctions* or *Second Price Auctions* (Vickrey Auctions). In both these types, the winner of auction is the bidder who bids the highest. However, the price paid by the winner differs in these two auctions. In the case of first price auctions, the winner pays the same price that it bid (and won). For second price auctions, winner pays the price which equals the bid value of second highest bidder. First price auctions (FPA) are widely believed to be fairer to everyone involved and companies like Google are shifting to FPA for auctioning advertisement spaces [Mor22]. In this work, we focus on first price sealed bid auctions.

Traditional auctions take place in presence of a trusted auctioneer – to whom the bidders provide their private bid values. However, the notion of a trusted party is difficult to realize in practice. In this digital age, with e-auctions being the norm, there is a need to replace such a trusted party with a secure protocol that does not leak any private bid values. Auctions also tend to be a high-stakes game for the participants, offering plenty of incentives for corruption. Therefore an auction protocol needs to ensure fairness (i.e., if one party learns the output, so do the other parties). Maliciously Secure Multi-Party Computation (MPC) offers a solution to this problem: the set of bidders, who may be mutually non-trusting and possibly corrupt, may simply run an MPC protocol to compute the maximum bid value. However, securing against malicious parties, that too using generic MPC, comes at a heavy cost: such solutions incur huge computational and communication overheads, making them inefficient for use in practice. Fortunately, for typical use-cases of auctions, safeguarding against malicious parties, who can deviate arbitrarily, is an overkill. Indeed, malicious strategies include those that parties may never adopt in real-life.

At Crypto 2009, Miltersen, Nielsen and Triandopoulos [MNT09], introduced the problem of building auctions that are secure against *rational* bidders. Such parties are modeled as a self-interested agents who care more about the monetary payoffs rather than learning information about bids of other agents. An important contribution of their work is that they put forth a novel notion of *information utility* and introduce game-theoretic framework that helps us capture the interplay between monetary and information utilitites. Formalizing this is not trivial because the utility of the same information may be different for different parties. Yet, this abstract concept, which cannot be concretized, must play a role for privacy-enhanced auctions in the rational setting and this makes it very challenging. Our work is inspired by this beautiful work.

In this work, we build auctions that are secure against *rational* parties who simply wish to maximize their utility. Our protocol is relevant in scenarios when parties can be disincentivized against certain kinds of "bad behavior" through monetary penalties. This is true, for example, in cases where parties register for the auction by paying a security deposit, which they will forego if cheating is detected. Prior work such as [MNT09,DGP22] have considered auctions for rational adversaries. We offer a detailed comparison of the efficiency improvements of our protocol in Section 1. Additionally, [DGP22] has no formal proof of security for rational adversaries. In fact, as the authors themselves point out, their work as well as that of [BHSR19] has some non-trivial leakage – the highest bidder learns information where he overtook the bid of the second highest bidder. We offer a rigorous, game-theoretic treatment of secure auctions and demonstrate that our protocol achieves *"Privacy Enhanced Computational Dominant Strategy Equilibrium"*, a notion that we introduce. Intuitively, this means, following the protocol is the preferred option for each party, irrespective of what other parties might choose to do.

Our Contribution. We present the first concretely efficient protocol for *First Price Auction* with guaranteed privacy that achieves *Computational Weakly Dominant Strategy Equilibrium* for rational parties without the need for any trusted setup while ensuring fairness. We have implemented our protocol in C++ with 1840 lines of code. We build upon OpenSSL and Boost open source libraries. Running our protocol on a commodity hardware (with Intel core i7 processor, 2.9 GH), with 10 bidders, 10 bit length bids resulted in 0.03MB communication and took 0.141s. We have shown that our protocol is concretely more efficient than other secure auction implementations, including ones using generic MPC protocols.

Table 1: Comparison of protocols

Protocol	**Security model**	**Equilibrium**
SEAL [BHSR19]	Passive [Non-trivial Leakage]	NA
FAST [DGP22]	Malicious [Non-trivial Leakage]	NA[a]
Protocol in [MNT09]	Rational	Computational Nash Equilibrium
Our Protocol	Rational	Computational Weakly Dominant Strategy Equilibrium

[a] The authors analyze incentives for a rational party to cheat, but do not provide a proof for rational security.

1.1 Technical Overview

The starting point of our work is the work of [BHSR19] which in turn uses the *Anonymous Veto Protocol*(AVP) due to [HZ06]. Originally, AVP was designed for computing the logical-OR of a set of 1-bit private inputs from different parties. This was enhanced to compute the maximum bid value (for auctions) in the work of [BHSR19]. We call this protocol Anonymous Bid Protocol (ABP) and it runs for l rounds, where l is the number of bits in the binary representation of bid values. Technically, in ABP, bits are written onto the bulletin board in the clear. However, when used in auctions, it is combined with a specific encoding scheme for the sake of privacy. Here, we give an overview of ABP with the encodings incorporated into it. The high level idea of the l-round ABP protocol [BHSR19] is as follows:

1. All messages are written onto a bulletin board.
2. Parties learn the highest bid value bit-by-bit. Let b_{ij} denote P_i's bit used in jth round. We say that a party P_i drops out of the race at the end of round j, if $b_{ij} = 0$ but $\exists i'$ such that $P_{i'}$ is still in the race and $b_{i'j} = 1$. Once a party P_i drops out of the race, it continues to participate in the protocol using only the dummy bid value of $b_{ij} = 0$ for all subsequent rounds.
3. INPUT SPECIFICATION. Parties specify their inputs in any round by encoding it using an encoding scheme with a specific structure. Encoding of a 1 bit is g^r for a random r and where g is the generator of an appropriately chosen group $\mathbb{G}$. P_i's encoding of 0 is a value with the following property: if all parties have a zero bit, then multiplying their encodings will result in 1.
4. HIGHEST BID EVALUATION. To evaluate, the parties simply multiply all the encoded bid values. (This can be done publicly by any of the parties which has access to the encoded bids.) If, in round j, the product is 1, then we conclude that all parties contributed only 0 in that round and in particular, the highest bidder P_{i^*} had a 0 in the index j i.e., $b_{i^*j} = 0$. If the product is not 1, then the parties conclude that the highest bit value in that round/index is 1. Note that this conclusion would indeed be correct, because of a) the property which encodings of 0 satisfy and b) the fact that as long as at least one party contributes g^r for a random r, the product of the encodings is unlikely to be 1.

At a high level, privacy follows from the Decisional Diffie Hellman (DDH) assumption. To use ABP to build secure auctions, [BHSR19,DGP22] do the following:

- All parties commit to their bid values.
- They run ABP using a bulletin board to communicate and at the end of which they learn the highest bid value.
- The highest bidder opens out his commitment to prove that he did have the highest bid.

– To secure against malicious adversaries, parties simply use NIZKs at every step in order to offer a proof of correct computation.

There are two issues with this protocol: first, as mentioned before, the use of NIZKs makes it inefficient and second, the highest bidder, through some offline computations, can learn information on the bid of the second highest bidder. The idea to learn this leakage is as follows: when a party contributes the encoding of a 1 bit in any round j, he can easily check if there is any other party who has a 1 in that round. To do so, instead of the encoding of 1 that he used, he computes an encoding of 0 and multiplies it with the rest of the encodings that have been written onto the bulletin board. (Note that he can do this offline.) If the product is equal to 1, he knows that all other parties have contributed encodings of 0. If not, he knows that the rest of the parties contributed with encodings of 1. The highest bidder, specifically, can use this technique to determine the round at the end of which the second highest bidder dropped out of the race and learn first few bits of second highest bid.

Preventing Leakage in [BHSR19,DGP22] Observe that it is the presence of the encodings on the bulletin board that allows all parties to learn the output (the winning bid value and identity of winner) in a publicly verifiable and fair manner. However, in order to prevent leakage, we need to prevent the parties from running offline computations with these encodings. This would in turn require us to ensure that the bulletin board does not directly contain encodings. At the same time, for the sake of efficiency, we need to avoid using generic MPC protocols. Our first idea is to designate a specific party as an *Evaluator* who, in addition to being a bidder, also handles the computation of the winning bid. To accomplish this, parties send their encodings to the evaluator. This can be accomplished by having the parties encrypt their encodings with the evaluator's public key and writing these on to the bulletin board. This will enable her to learn the per-round outputs (i.e., the highest bid value for that round) and reveal it to the other parties. One obvious issue which arises is that this violates fairness but we can enforce by imposing monetary penalties if that happens, thereby making that strategy irrational for the evaluator .

A bigger challenge is that if the evaluator has all the encodings, then in a case when the evaluator herself has the highest bid, she can learn the exact same leakage as before. The following observation is critical to our protocol: the evaluator does not need to learn the encodings when she is contributing a 1 bit herself as she can directly set the output of that round to be 1. We therefore need to ensure that she does not learn the encodings in such cases. At the same time, the protocol cannot leak to other parties that the evaluator has a 1 bit. *Oblivious transfer (OT)* to the rescue! Parties run a maliciously secure OT protocol with the evaluator wherein, if the evaluator 's contributing bit in a round is 0, she learns the bit encodings as before and if it is 1, she learns cipher texts of dummy messages. This introduces a further problem: the evaluator could cheat and ask for real encodings even though she has a 1 and is still in the race. While this can easily be solved using NIZKs, the main challenge is in figuring out how the evaluator can prove correctness of computation *without* relying on NIZKs. One way would be to insist that she "open out" all the randomness used in her OT computations (as well as the commitment to her input) so that the other parties can verify correctness. Revealing her own input is a violation of her privacy unless she has the highest bid. While this might seem like a deadlock situation, we argue that it is not. The key idea of this protocol is that the evaluator only learns information if she is the highest bidder. If she is not the highest bidder, she learns nothing other than the winning bid by cheating and it is fine for the other parties to not detect this cheating. In other words, she does not need to give a proof when she is not the highest bidder. (We stress that in the entire protocol, we will have several cheating strategies which may go undetected; this won't be a problem – we will argue that in all such strategies too, the parties will learn nothing extra and have no incentive to cheat.) To sum up, when the evaluator is the highest bidder, we will have the evaluator offer a proof of correct computation by revealing the randomness used to compute the OT messages as well as her input. This does not leak any information because the evaluator's bid is the highest and is therefore not a secret!

As it turns out, even with this modification, the highest bidder could potentially learn information by deviating from the protocol. Consider a case when a party with the highest bid uses a 0 bit instead of a 1 bit in some round j. For the rest of the rounds, he follows the protocol as though he did not deviate. Now consider a scenario where he actually wins – this means that there was another party who had a 1 in round j. This is auxiliary information that he is not authorized to learn. Intuitively, what we need is a way for parties to give a proof of correct computation, in the event that they win. However, this is complicated by the fact that the evaluator does not even use the true input of the party in cases where she herself contributes a 1. Added to this, we also need to deter the evaluator from cheating. All of this results in a few more extensions to the protocol which we describe informally in the overview below.

Finally, it may also be possible that when the evaluator announces the winning bid, no one comes forward to claim the bid. In such a case, parties need to give a proof of not winning. Note that here too

we avoid the use of NIZKs by carefully analyzing the ABP protocol and providing an alternate proof that will help detect the deviating party. Specifically, we observe that there exists a certain round, which we call as *last decider round* in which the winning party alone uses a 1-bit code and all other parties use 0-bit codes. This means, an honest losing bidder can prove that it indeed used a 0-bit code during the last decider round. This important observation leads us to design proofs of not winning when the winner fails to turn up.

To obtain the final protocol, we carefully consider these and many other subtle adversarial strategies. We refer the reader to Section 3. For now, we present a high level overview of the protocol.

Protocol Overview

1. Set-up Phase We assume that all P_i who wish to participate in the auction, register by paying a deposit D.
2. Each party P_i has as input its private valuation of auctioned item v_i, bid value b_i.
3. P_i receives the public parameters pp.
4. Party P_i commits to its bid value as well as to the hash of the randomness he will use in the rest of the protocol. (This will be used to verify correctness of P_i's computations, in the event that he claims to have the winning bid.)
5. One of the parties is chosen to be the evaluator and we denote it by P_e.
6. Computation Phase The protocol proceeds in rounds and in each round parties run the ABP protocol as explained earlier. This roughly translates to running an OT protocol with the evaluator to reveal the encoding of bit it chooses to *contribute* in that round (or the encoding of a dummy bit, depending on the evaluator 's input). We use the terminology "contributing to a round" to stress the point that the party may not input its true bid value for that index, if it is no longer in the race. At the end of this phase, the evaluator announces the winning bid.
7. If no one comes to claim the winning bid, all parties offer a proof of not winning.
8. A party, P_w, claiming to have the winning bid, needs to give a proof of winning. First, he opens the commitment to his bid value which matches the winning bid. Additionally, he engages in a protocol with the evaluator to convince the evaluator that he did play honestly. Roughly this consists of the following steps:
   (a) P_w opens his commitments to reveal his bid as well as the hash of the randomness. He also shares the randomness to enable verification of the hash value.
   (b) The evaluator uses her own randomness which she uses to generate the OT first message to check the consistency of the encodings. Informally, the evaluator verifies the following statement for each round: "When I contributed a 0 bit, the prover did send me his true encoding which is consistent with the bit in that index. When I contributed a 1 bit, the prover's second message is computed correctly as a function of the encoding of his true input, encoding of his dummy input, the randomness he committed to, as well as my OT first message." This is easy to accomplish given that the P_e has the randomness that P_w has used to generate the OT second message as well as the encodings.
9. Whenever a party's deviation is detected, the protocol terminates and the party forgoes their deposit.

Security In order to argue security, we note that a rational party may see value in a) increasing his monetary utility by winning the protocol and b) learning information about other players' inputs. To capture the latter, we use the term *information utility*, as introduced in [MNT09]. We introduce the notion of *Privacy Enhanced Computational Weakly Dominant Strategy Equilibrium* for analyzing the privacy concerns of rational parties. This equilibrium states that as long as parties value monetary utility much more than information utility, there is no incentive for them to deviate from the protocol. Next, for parties that do not deviate from the protocol, we argue privacy using the simulation paradigm.

While there have been works on cryptographic protocols achieving *Computational Nash Equilibrium* (e.g. in [MNT09,MR12,DHR00]), to the best of our knowledge, ours is the first work that is able to establish *Privacy Enhanced Computational weakly Dominant Strategy Equilibrium* for first price auction protocols. This equilibrium notion can be of independent interest.

1.2 Related Work

Rational Secure Computation. The work of Dodis, Halevi and Rabin [DHR00] initiated a line of work on capturing game-theoretic notions of incentives in cryptographic definitions. The work of Halpern and Teague [HT04] considered secret sharing and secure computation in the rational setting where parties are rational with a utility function that is curious-then-exclusive: parties primarily prefer to learn the output, and secondarily, if they learn the output, have as few other parties learn it as possible. They define a solution concept that is a notion of Nash equilibrium. Further works define more variants of equilibrium like computational versions, modeling collusion etc. [ADGH06,GK06,Hal08,KN08,HP10]. Note that while a Nash equilibrium ensures that following the protocol is the optimal strategy when all other protocols do follow the strategy, it does not say anything in scenarios where parties are likely to deviate. A dominant strategy equilibrium, on the other hand, does not care for other people's strategies: following the protocol is indeed the most preferred strategy for a party regardless of what other people's strategy might be. Biçer, Yildiz and Küpçü make use of *Weakly dominant strategy* for coalitions in their work [BYK21] to develop the notion of *m-stability* which offers threshold security against a coalition of size m. Another rich line of work in rational cryptography is in the Rational Protocol Design (RPD) framework introduced by Garay, Katz, Maurer, Tackmann and Zikas [GKM+13] and employed subsequently in [GKTZ15,BGM+18]. In RPD, a two-party game is considered 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. RPD makes analysis of collusions, adaptive corruptions etc. easier, and also provides compositional guarantees. In [MR14], Micali and Rabin discuss prevention of collusion among bidders, participating in Vickrey auctions. In our work, modeling incentive-driven behaviour of mutually distrustful parties suffice, and we leave extending our protocol to the RPD framework and Vickrey auctions to future work.

Covert Secure Computation. A different line of work on covert adversaries incorporates some types of incentives into cryptographic definitions. Aumann and Lindell [AL07] proposed the notion of covert adversaries that are in between standard semi-honest and malicious adversaries: these are parties who will cheat but only as long as they do not get caught cheating. This "fear" of getting caught is used to build more efficient protocols [AO12,Lin13]. This model is also captured by the rational model by defining the utility to be negative when an abort happens in an identifiable way.

Rational Secure Auctions. In the context of auctions, our work is closely related to three works. The work of Miltersen, Nielsen and Triandopoulos [MNT09] defines a rational security framework for auctions and introduce the notion of *Privacy enhanced ϵ-Nash Equilibrium*. We use the notions of monetary and information utility as in [MNT09], but put forth a different solution concept. Additionally, the protocol of [MNT09] uses generic MPC techniques secure against active corruptions. In our work, however, we take advantage of parties being rational, and not arbitrarily malicious, to construct an efficient protocol. Our protocol is relevant in scenarios where (a) monetary penalties can be imposed on parties whose cheating is detected and (b) parties have access to a bulletin board. Both these assumptions are easily implementable in practice. Many auctions require parties to register with a fee anyway. The latter can be implemented with a simple webpage. Needless to say, for blockchain applications of auctions, the blockchain itself is a bulletin board, and one can simply use a smart contract for managing the deposits. On-chain auctions have applications in bootstrapping a virtual ASIC blockchain system [GOTZ21], and in auction-based minting mechanisms [DDM+20].

The work of Bag, Hao, Shahandashti, and Ghosh [BHSR19] construct an auction scheme called SEAL that works without any trusted party, and secure against passive adversaries. David, Gentile, and Pourpouneh propose FAST [DGP22] that builds on SEAL to provide security against active adversaries. While FAST also informally argues that the protocol enforces honest behavior among rational bidders, a formal rational security analysis is missing. Like SEAL and FAST, our protocol uses the *Anonymous Veto Protocol* (AVP) introduced in [HZ06] as a building block. Our protocol is more efficient than both SEAL and FAST, and is proven to be secure in the presence of rational adversaries.

1.3 Comparison with other approaches

We now compare with other straightforward approaches, highlight why they do not work and their shortcomings.

Timed primitives. Timed commitments [BN00,MT19]: These are an extension to standard commitments to have a forced opening phase wherein one can solve a time lock puzzle in order to "open" the commitment. It may appear that these automatically give sealed bid auctions achieving fairness. However,

there are several issues with this approach. First, they offer privacy only for a certain period of time. While this may be sufficient for some applications, our auctions are designed to satisfy long-term privacy.

Covert secure MPC. An MPC protocol is said to be secure against covert adversaries if honest parties are guaranteed to detect malicious behavior. If we impose monetary penalties on such detected parties, does it help achieve our goal? It does not. First, such protocols only detect cheating with a constant probability, and when cheating goes undetected, the adversary learns private inputs. Increasing the detection probability to be all but negligible, will result in higher communication cost in existing covert secure protocols [GMS08]. In the vanilla covert mode (without identifiable abort) [SSS21], the detection is not necessarily unanimous among honest parties. Therefore, it is unclear how the penalty is imposed, and who it is distributed to. One way to fix this is by requiring protocols with identifiable abort/public verifiability, but these are achieved through the use of expensive TLPs.

However, in general it is non-trivial to modify protocols based on timed commitments and covert secure MPC to achieve privacy preserving auctions.

Using NIZKs. Using NIZKs on top of the ABP protocol as described in Section 1.1 makes the protocol inefficient. The languages that need NIZKs (like composition of proofs of knowledge of discrete log or opening of a Pedersen commitment) admit Sigma protocols. However, concretely, since a NIZK proof needs to be sent per party per round, this adds to the communication overhead (roughly, $2n\ell|\mathbb{G}|$ where n is the number of parties, ℓ is the bit length of the bid and $|\mathbb{G}|$ is the size of the group element used by the protocol). Prior work on auctions, FAST [DGP22] and SEAL [BHSR19] do make use of NIZKs in their protocols, but are inefficient. Moreover, even with NIZKs, the resulting ABP-based protocol admits leakage on the second highest bid. Our approach avoids this overhead, results in a reduced number of public key operations thus giving better concrete efficiency, and provides full privacy.

2 Preliminaries

Notation. We denote the security parameter by λ. Let $\mathbb{G}$ be the description of the group of order q, and g, g_1, h be the generators of $\mathbb{G}$. 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 distributions.

2.1 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 1 (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 set of possible actions Γ_i along with a utility function U_i are specified.*

Depending on its private inputs, each party P_i uses strategy $\pi_i \in \Gamma_i$ where Γ_i is the space of strategies available for party P_i, during the game. We also assign utility functions $U_i(\pi_i, \pi_{-i})$ to each party P_i. These functions represent the perceived utility of game for the party. We say that the party P_i *prefers* the action $\pi_{\mathbf{i}}$ over action $\pi_{\mathbf{i}}{}'$ if and only if $U_i(\pi_{\mathbf{i}}) > U_i(\pi_{\mathbf{i}}{}')$.

Definition 2 (Dominant Strategy [Kat08]). *Given a normal form game:*

$$\{\{\Gamma_i\}_{i=1}^n, \{U_i\}_{i=1}^n\}$$

we say $\pi_i \in \Gamma_i$ is a Dominant Strategy *if $U_i(\pi_i, \pi_{-i}) > U_i(\pi_i', \pi_{-i})$ for all $\pi_i \neq \pi_i' \in \Gamma_i$ and for all $\pi_{-i} \in \Gamma_{-i}$.*

Such a strategy π_i guarantees that the party P_i can accrue best utility among all strategies available to it. In the above case, π_i' is also termed as *Dominated Strategy*. Dominated strategies are typically avoided by the rational parties, whereas dominant strategies are pursued.

We also have a weaker notion of *Dominant Strategy* known as *Weakly Dominant Strategy*.

Definition 3 (Weakly Dominant Strategy [Kat08]). *Given a normal form game:*

$$\{\{\Gamma_i\}_{i=1}^n, \{U_i\}_{i=1}^n\}$$

we say $\pi_i \in \Gamma_i$ is a Weakly Dominant Strategy *if $U_i(\pi_i, \pi_{-i}) \geq U_i(\pi_i', \pi_{-i})$ for all $\pi_i \neq \pi_i' \in \Gamma_i$ and for all $\pi_{-i} \in \Gamma_{-i}$.*
In addition, there exists some $\pi_{-i} \in \Gamma_{-i}$ such that $U_i(\pi_i, \pi_{-i}) > U_i(\pi_i', \pi_{-i})$.

As can be seen, a *Weakly Dominated Strategy* can realize the same utility as the Dominant Strategy for certain strategies.

Definition 4 (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* $\pi = (\pi_1, \ldots, \pi_n) \in \Gamma$ *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 is present, 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 in the game, the chosen strategy profile is an equilibrium. In our case, we show that every party has a *weakly* dominant strategy – which is to follow the protocol.

2.2 Building Blocks

Here we describe some of the key building blocks that we make use of in our protocol.

Definition 5 (Commitment scheme). *Let* M*,* C*,* R *be the message space, commitment space and randomness space respectively.*
A commitment scheme consists of a tuple ($\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}$*) of PPT algorithms where:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *generates public parameters pp*
- $\mathsf{Commit}(\mathsf{pp}; m) \to (c; r)$ *takes a message* $m \in \mathsf{M}$*, randomness* $r \in \mathsf{R}$ *and outputs a commitment* $c \in \mathsf{C}$
- $\mathsf{Open}(\mathsf{pp}, c, m, r) \to B \in \{0, 1\}$ *checks if the commitment c opens to the message m*

The security of commitment scheme guarantees two properties: *hiding* and *binding*. Informally, the hiding property guarantees that for any two messages m_0, 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. We use the *Pedersen Commitment Scheme* that is computationally binding and perfectly hiding.

Definition 6 (Pedersen Commitment Scheme [Ped92]). *The Pedersen Commitment Scheme* Com *instantiates the Commitment Scheme defined above as:*

- $c = \mathsf{Com}(m, r) = g^m h^r$ *where* $m \in \mathbb{Z}_q$*,* $r \leftarrow_R \mathbb{Z}_q$ *is chosen uniformly at random and* $c \in \mathbb{G}$*.*
- $\mathsf{Open}(c, m, r) \to B \in \{0, 1\}$ *verifies if c is indeed the right commitment for message m with opening randomness r.*

The *Vector Pedersen Commitment* Scheme allows one to commit to $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ as follows: given generators $h, g_1, \ldots, g_n$, such that the discrete logarithm between any two generators is unknown, $c = \mathsf{Com}(\mathbf{m}, r) = h^r \prod_{i=1}^n g_i^{m_i}$.

Oblivious Transfer. We make use of Oblivious Transfer for securely sharing messages between parties. Let M, R be the message space and randomness space respectively. An OT protocol proceeds as follows for two PPT parties R and S:

- $\mathsf{OT.R}_1(\alpha, \beta) \to (otr_1, state)$ is invoked by R with inputs $\alpha \in \{0, 1\}$, randomness $\beta \in \mathsf{R}$. otr_1 is the first message sent by R to S. *state* denotes the state of protocol.
- $\mathsf{OT.S}(m_0, m_1, \gamma) \to ots$ takes messages $m_0, m_1 \in \mathsf{M}$, randomness $\gamma \in \mathsf{R}$ and outputs message to be sent by S to R.
- $\mathsf{OT.R}_2(ots, state) \to m_\alpha$ uses the message from S and internal state to retrieve the message m_α.

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

Collision Resistant Hash Function. A collision resistant hash function (CRH) is a function family for which it is hard to find inputs that map to the same output. More formally, a family $\mathcal{H}_k$ is a CRH if for every PPT $\mathcal{A}$, $\Pr_{\mathsf{hash} \leftarrow \mathcal{H}_k}((x_1, x_2) \leftarrow \mathcal{A}(\mathsf{hash}) | x_1 \neq x_2, \mathsf{hash}(x_1) = \mathsf{hash}(x_2)) \leq \mathsf{negl}(\lambda)$ where each hash maps ℓ bit strings to k bit strings.

Bulletin Board. The *Bulletin Board* (BB) is an abstraction for authenticated broadcast channel with memory. In our protocol, parties can write messages on to the BB for public consumption. The BB is expected to satisfy following properties: (i) Every message written on the BB is associated to a party and is readable by all other parties. (ii) The messages written on BB are immutable. A bulletin board can be realized using blockchains. However, for several real-world use cases of auctions – tech auctions, sporting auctions, for example – we might be willing to place a minimal assumption on the availability of a bulletin board (a website, for example).

3 Auction Protocol

We first describe our security model. We then present some building blocks used by the protocol followed by a detailed protocol specification.

3.1 Security Model

We consider rational PPT parties that are not viewed as being either honest or corrupt; instead, they are simply rational and are motivated by some utility function.

We are interested in strategies that are self-enforcing – i.e., each party chooses such a strategy that there is no incentive to deviate from it. In other words, the choice of strategies is an equilibrium. We seek a *Dominant Strategy Equilibrium* for computationally bounded parties for non-adaptive strategies. However, such a preference to follow the protocol among parties, in itself does not address privacy concerns of the parties. We need more guarantees from the protocol for addressing privacy. For this, we introduce the notion of *Privacy Enhanced Computational Weakly Dominant Strategy Equilibrium*, inspired by the work of [MNT09].

Definition 7 (Privacy Enhanced Computational Weakly 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. Let $\pi_i \in \Gamma_i$ be the strategy for P_i of following Π. Let $\pi_i' \in \Gamma_i$ be an arbitrary efficiently computable strategy.*

We say that the protocol Π is a Privacy Enhanced Computational Weakly Dominant Strategy Equilibrium *if the following hold with probability $(1 - \mathsf{negl}(\lambda))$, where λ is the security parameter.*

1. *Π is a* Weakly Dominant Strategy Equilibrium*; i.e.,*

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

   for all arbitrary efficiently computable $\hat{\pi}_{-i}$.

2. *For every P_i, there exists a simulator Sim_{P_i} such that the view of P_i in a real execution of the protocol is computationally indistinguishable from the output of the simulator:*

$$\mathsf{View}^{\Pi}_{P_i,\mathsf{real}} \approx_c \mathsf{Sim}^{\Pi}_{P_i}$$

where $\mathsf{View}^{\Pi}_{P_i,\mathsf{real}}$ is the random variable of the transcript of P_i in protocol Π.

3.2 Anonymous Bidding Protocol

In constructing our protocol, we use the *Anonymous Bidding Protocol* (ABP) to compute the highest bid value which is a variation of the Anonymous Veto Protocol (AVP) first described in [HZ06] and later used with modifications in [BHSR19] and [DGP22].

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

- Each bidder P_i participates in ABP by inputting one bit from their bid value at a time. This continues until he is in the race. If in a round, there is a bidder who inputs a 1 while the party P_i has 0 in that index, he drops out of the race at the end of that round.
- A party who has lost out on the race, continues to participate in the protocol but only contributes a 0 to the rest of the computation.
- 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 winning bit. i.e.,

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

Observe that, decider round j has the corresponding winning bit $b_{wj} = 1$.

– Winning bid b_w is computed as $b_w = b_{i1}||\cdots||b_{il}$

We make use of this protocol for computing the highest bid value. Although the ABP protocol has been described above using raw bits, for the actual computation, raw bits are not used. Instead we make use of encoded bits that hide the actual bit values – thus preserving privacy.

3.3 Bit Encoding Scheme

In order to ensure the privacy of the bits used during computation, we require the bits to be encoded such that no PPT party can distinguish between the encoding of 0 and encoding of 1. For this we make use of Bit Encoding Scheme (BES). This scheme is adopted from [BHSR19]. It follows from the DDH assumption holding in the group $\mathbb{G}$ that the distributions of 0-bit code and 1-bit code are computationally indistinguishable. The encoding and corresponding computation on the coded bits are performed as follows:

1. Each bidder P_i, $i \in [n]$ allocates private keys $x_{ij}, r_{ij} \leftarrow_R \mathbb{Z}_q$, $i \in [n], j \in [l]$. 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 : $B_{ij} = \mathsf{BESEncode}(d_{ij})$ where

$$\mathsf{BESEncode}(d_{ij}) = \begin{cases} \text{0-bit code}: & Y_{ij}^{x_{ij}} & \text{if} & d_{ij} = 0 \\ \text{1-bit code}: & g^{r_{ij}} & \text{if} & d_{ij} = 1 \end{cases}$$

4. The jth winning bit is computed as the logical-OR of individual bidding bits b_{ij} for the jth position: $b_{wj} = \bigvee_{i=1}^{n} b_{ij}$. This is computed using the bit codes as follows:

$$b_{wj} = \begin{cases} 0, & \text{if} \quad \prod_{i=1}^{n} B_{ij} = 1 \\ 1, & \text{if} \quad \prod_{i=1}^{n} B_{ij} \neq 1 \end{cases}$$

Legend:

– $g \in \mathbb{G}$ is a generator of group $\mathbb{G}$ of prime order q where DDH assumption holds.

– Each bid value b_i for party $P_i, i \in [n]$ is represented in binary using l bits.

We refer the reader to [BHSR19] for proofs establishing the indistinguishability of the distributions of 0-code and 1-code which follows from DDH.

3.4 Our Protocol

In this section, we describe the protocol Π for running a First Price Auction. For the sake of simplicity, we assume that the bid values are all distinct. However, we can easily extend our protocol to include a simple tie-breaking mechanism. We also make use of fixed value of security deposit amount of D for all parties. Since most practical auctions have a *reserve price*, that is the minimum bidding amount, the deposit D can be chosen to be equal to the *reserve price* of the auction. Of this deposit, $D/2$ is used exclusively in the last phase of the protocol where the potential winner (who claims to have the winning bid) engages with the evaluator to verify that he does indeed have the winning bid. The remainder of the deposit is something that any party may forfeit if they are caught cheating in other parts of the protocol.

Protocol Specification The protocol for running first price auctions is specified below. We make use of a maliciously secure OT protocol $\Pi_{OT} : (\mathsf{OT.R}_1, \mathsf{OT.R}_2, \mathsf{OT.S})$.

We note that selecting public parameters(pp) in our protocol does not require any central authority or use of a separate protocol. Our pp – comprising of a DDH hard group $\mathbb{G}$, generator g – can be generated using off-the-shelf parameters as recommended by NIST (e.g. secp256k1 elliptic curve). Other generators

of the group can be chosen by hashing the generator g.

Setup Phase:

1. Each party P_i is initialized with its private valuation of auctioned item v_i, bid value b_i.
2. Each party P_i registers for the auction by depositing its security deposit D. In turn P_i receives the public parameters $\mathsf{pp} = (q, \mathbb{G}, g, g_1, h)$.
3. The party P_i samples the private keys $x_{ij}, r_{ij} \leftarrow_R \mathbb{Z}_q$, $j \in [l]$
4. P_i generates a CR hash for the concatenated private randomness
   - For $i \neq e$, $\rho_i = x_{i1}||\cdots||x_{il}||r_{i1}||\cdots||r_{il}||\gamma_{i1}||\cdots||\gamma_{il}$. Note that x_{ij}s and r_{ij}s are used to encode bit 0 and bit 1, respectively. γ_{ij}s are used as randomness for generation of OT messages by OT sender P_i.
   - For $i = e$, $\rho_i = x_{i1}||\cdots||x_{il}||r_{i1}||\cdots||r_{il}||\beta_{11}||\cdots||\beta_{nl}$. Again, x_{ij}s and r_{ij}s are used to encode bit 0 and bit 1, respectively. β_{kj}s are used as randomness for generation of OT messages by OT receiver (evaluator P_e) while interacting with the party P_k.
   - $H_i = \mathsf{hash}(\rho_i)$
5. Party P_i chooses $R_i \leftarrow_R \mathbb{Z}_q$. Computes commitment (as per Definition 6) $c_i = \mathsf{Com}(b_i, H_i, R_i) = g^{b_i} g_1^{H_i} h^{R_i}$. These commitments are written to BB.
6. Writes the public keys $X_{ij} = g^{x_{ij}}$ for $1 \leq j \leq l$ to BB.

Computation Phase

1. For $j \in [l]$, each party P_i chooses his contributing bidding bit d_{ij} following ABP as specified in §3.2. P_i generates the bit codes $B_{ij} \leftarrow \mathsf{BESEncode}(d_{ij})$ (with either x_{ij} or r_{ij} as appropriate).
2. P_e plays the role of the Receiver in pair-wise OT with all other parties $P_i, i \neq e$, to obtain the bit codes as follows for rounds $j = 1$ to l:
   - P_e sets $\alpha_j = d_{ej}, \beta_{ij} \leftarrow_R \mathbb{Z}_q$ and invokes $\mathsf{OT.R}_1(\alpha_j, \beta_{ij})$
   - OT sender P_i sets $M_{ij}^{(0)} = B_{ij}$, $M_{ij}^{(1)} \leftarrow_R \mathbb{G}$. Samples $\gamma_{ij} \leftarrow_R \mathbb{Z}_q$. P_i invokes $\mathsf{OT.S}\left(M_{ij}^{(0)}, M_{ij}^{(1)}, \gamma_{ij}\right)$ to obtain $C_{ij}^{(0)}, C_{ij}^{(1)}$, which are written to BB.
   - When $\alpha_j = 0, P_e$ retrieves the bit code $B_{ij}, i \neq e$ from OT using $\mathsf{OT.R}_2(C_{ij}^{(0)})$, computes the bit b_{wj} as mentioned in BES description §3.3 and writes the same to BB. If the bit $b_{wj} = 0$, P_e also writes the bit-codes for each bidder to the BB as a proof of correct computation of 0-bit.
   - When $\alpha_j = 1$, P_e writes $b_{wj} = 1$ to BB.

   After l rounds, the protocol proceeds to **Verification Phase**.
3. If any party P_i fails to write its messages or if P_e doesn't output the bit b_{wj} to BB within τ time units, they are considered to have aborted and protocol proceeds to terminate and the deposits are redistributed.
4. After l rounds, each party constructs the value of winning bid as $b_w = b_{w1}||b_{w2}||\ldots||b_{wl}$.

We denote the evaluator by P_e in the specification.

Verification Phase – Winner verification

1. The party P_w claims auction by opening its commitment. P_w writes R_w to BB.
2. If winner is not the evaluator , P_w opens its OT sender randomness γ_{wj} for all $j \in [l]$ and writes the same to BB.
3. P_w writes all its private keys x_{wj}, r_{wj} for rounds $j \in [l]$ to BB.
4. Evaluator verifies validity of private randomness:

$$H_w \stackrel{?}{=} \mathsf{hash}\left(x_{i1}|| \cdots ||x_{il}||r_{i1}|| \cdots ||r_{il}||\gamma_{i1}|| \cdots ||\gamma_{il}\right)$$

5. Evaluator uses the OT sender randomness γ_{wj} and $C_{ij}^{(0)}$ to retrieve the bit codes used by P_w during the computation phase.
6. Evaluator also uses the private keys shared by P_w to compute the bit codes and checks if they match with retrieved bit codes. For each $j \in [l]$:

$$\text{if } b_{wj} = 1 : B_{ij} = g^{r_{ij}}$$

$$\text{if } b_{wj} = 0 : B_{ij} = Y_{ij}^{x_{ij}}, \text{where } Y_{ij} = \frac{\prod_{k=1}^{i-1} X_{kj}}{\prod_{k=i+1}^{n} X_{kj}}$$

and X_{kj} is public key for round j posted by parties P_k.

If there is a match, confirms that P_w is the winner.

7. If there is discrepancy in some round j, Evaluator flags the same against P_w. In case there is any rebuttal from P_w, then evaluator opens its OT sender randomness β_{wj} during round j to prove that P_w is indeed cheating. In such a case, P_w loses its entire deposit D such that, $D/2$ is redistributed among remaining parties and $D/2$ is paid out to evaluator to compensate for its loss of information during rebuttal.
8. In case evaluator P_e is the winner, P_e opens its OT first message randomness β_{ij} for all $i \in [n] \setminus \{e\}$ to prove that winning bits are same as those used as choice bits of OT.

Note that if cheating is detected by any of the parties, the protocol terminates and the deposits are appropriately redistributed. If there is no cheating, the protocol terminates and P_w wins the auction.

Verification Phase – No Claim verification

1. If P_w doesn't announce itself, each party provides *proof of not winning* for the last decider round j to prove that it has used 0-bit code in that round.
   (a) Each party P_i writes its secret key x_{ij} used to generate the 0-bit code to BB.
   (b) Evaluator P_e opens its OT first message to prove that it has used choice bit as 0 during round j. P_e also opens its OT receiver randomness β_{ij} for each party during round j.
   (c) Party $P_i, i \neq e$ opens randomness used for sending OT messages, γ_{ij}. This helps to reconstruct the bit code. These values are written to BB.
   (d) P_i's proof can be verified as: Compute $M_{ij}^{(0)}$, use the secret key x_{ij} to generate the 0-bit code B_{ij} and check if $B_{ij} \stackrel{?}{=} M_{ij}^{(0)}$. P_i is deemed cheating if the equality doesn't hold.
2. If every party P_i is able to send correct proof of not-winning, this means, the computed value b_w is incorrect. This can happen only when P_e has cheated. Thus, in such a case, P_e is considered to be cheating. The protocol terminates and the deposits are appropriately redistributed.

Fig. 1: Specification for First Price Auction Protocol Π

4 Correctness of Protocol

In this section, we will show that the protocol specified in Fig. 1 correctly computes the winning bid value. The correctness proof is adopted from [BHSR19]. However, we have simplified the argument and notation as per our protocol. First we show that each round of the protocol correctly computes the logical-OR of the bits supplied by the individual parties. Then we establish that the combined computation of all rounds indeed computes the highest bid.

Lemma 1. *For each round $j \in [l]$, the protocol specified in Fig. 1 correctly computes the logical-OR of the bits supplied by each bidder for that round.*

Proof. Consider an arbitrary round j of the protocol and an arbitrary party P_i. At the beginning of the round, each party P_i and the evaluator compute d_{ij} to be used during the ABP computation. The computation is done as follows:

- If $d_{ij} = 1$, P_i samples $r_{ij} \leftarrow_R \mathbb{Z}_q$.
  Generates the 1-bit code as $B_{ij} = g^{r_{ij}}$.
- If $d_{ij} = 0$, P_i computes $Y_{ij} = \dfrac{\prod_{k=1}^{i-1} X_k}{\prod_{k=i+1}^{n} X_k}$.
  Generates 0-bit code as $B_{ij} = Y_{ij}^{x_{ij}}$.

Each round starts with the evaluator initiating OT with the bidders. Recall that the evaluator uses the bit d_{ej} as per the ABP to be its choice bit. Accordingly, we will consider two cases.

1. Evaluator has choice bit as 1:
   P_e in this case recovers a random group element as P_i's message for each $i \in [n-1]$. Since $d_{ej} = 1$, P_e writes $b_{wj} = 1$ on to BB.
2. Evaluator has choice bit as 0:
   P_e recovers the bit codes from each party and computes the winning bit as:

$$b_{wj} = \begin{cases} 1, \text{if } \prod_{i=1}^{n} B_{ij} \neq 1 \\ 0, \text{if } \prod_{i=1}^{n} B_{ij} = 1 \end{cases}$$

   Observe that if there exists at least one party P_i which has $d_{ij} = 1$ and thus shares a 1-bit code B_{ij} with evaluator , the product $\prod_{i=1}^{n} B_{ij} \neq 1$ with overwhelming probability. Thus, when at least one of the parties has a 1 bit, the winning bit is computed as 1 which is indeed a logical-OR of the bits used by bidders in jth round. On the other hand, if every party shared a 0-bit code during the round j, then $\prod_{i=1}^{n} B_{ij} = 1$ and hence the output bit is computed correctly as 0.

Theorem 1. *The auction protocol specified in Fig. 1 computes the highest bid value correctly when all parties follow the protocol.*

Proof. Let P_w be the winning party with winning bid b_w. Let the l bits output by the protocol be $(win[1], \ldots, win[l])$. We now show that winning bit in jth position is same as the computed bit in jth round. I.e., $win[j] = b_{wj} \;\; \forall j \in [l]$.

Let m be the first decider round of the computation. According to the protocol, $win[m] = \bigvee_{i=1}^{n} b_{im} = 1$ and for all $j \in [1, m-1], win[j] = 0$. Note that this exactly corresponds to the first m bits of the winning bid as well. I.e., the first $(m-1)$ bits of winning bid value are all 0 and mth bit is 1. If this was not the case, then $b_{wj} = 1$ for some $j < m$. Then P_w would have used 1-bit for computation of logical-OR during jth round, thus obtaining $win[j] = b_{wj} = 1$. Then, j would be the first decider and not m contradicting our assumption above.

For the subsequent iterations, all bidders who bid 0-bit codes during the mth iteration would continue bidding only 0-bit codes for rest of the protocol. Hence, they would not contribute to the winning bid value. On the other hand, all bidders who bid 1-bit code during mth iteration would bid their actual bid value for other iterations. In particular, winning bidder would always bid 1-bit code during each of the decider rounds.

Let d_{ij} denote the bid value used by the bidder P_i during jth iteration. Then we have

$$d_{ij} = \begin{cases} 0, & \text{if } b_{im} = 0 \\ b_{ij}, & \text{if } b_{im} = 1 \end{cases}$$

Thus, we need to prove that in any iteration $j > m$ if $win[j] = \bigvee_{i=1}^{n} d_{ij} = 1$, then $b_{wj} = 1$ and if $win[j] = \bigvee_{i=1}^{n} d_{ij} = 0$, then $b_{wj} = 0$. We prove it by method of induction on the iteration number j after m.

For the base case, consider the iteration $j = m+1$. In this round evidently, $\bigvee_{i=1}^{n} b_{i,m+1} = 1$ whenever $b_{w,m+1} = 1$. On the other hand, when $b_{w,m+1} = 0$, it must be the case that all other bid bits are also zero and hence $\bigvee_{i=1}^{n} d_{i,m+1} = 0$. If this is not the case, there exists a party P_k who has $b_{k,m+1} = 1$ and hence with higher bid than P_w – which contradicts our assumption.

For the induction step, let us assume that the assertion holds for all $j \in [1, k-1]$, for some arbitrary $m+1 \leq k \leq l$. We need to prove that $win[k] = b_{wk}$. Observe that for iteration k,

$$\begin{aligned} win[k] &= \bigvee_{i=1}^{n} d_{ik} \\ &= \left(\bigvee_{i \in [n] \setminus \{w\}} d_{ik} \right) \bigvee d_{wk} \\ &= \left(\bigvee_{i \in [n] \setminus \{w\}} d_{ik} \right) \bigvee b_{wk} \end{aligned}$$

Here we have used $d_{wk} = b_{wk}$ since winner P_w would have used 1-bit code for the round m and other decider rounds till $(k-1)$th round. The $win[k]$ trivially evaluates to 1 whenever $b_{wk} = 1$. So let us consider the case when $b_{wk} = 0$.

Claim. $b_{wk} = 0 \implies \bigvee_{i \in [n] \setminus \{w\}} d_{ik} = 0$

Proof. Towards contradiction, assume that $b_{wk} = 0$, but $\bigvee_{i \in [n] \setminus \{w\}} d_{ik} = 1$.

This means, $\exists P_s, s \in [1, n]$ such that $b_{sj} = 1$ and P_s has been bidding 1-bit codes during all decider rounds. This implies that P_s's bid is higher than that of P_w. This follows because, from the claim's assertion, kth bit for b_w is 0 whereas kth bit of b_s is 1. This means, P_w is not the winner of the bid, but P_s is the winner. This is a contradiction, since we have assumed that P_w is the winner of bid. Thus it follows that $b_{wk} = 0 \implies \bigvee_{i \in [n] \setminus \{w\}} d_{ik} = 0$.

This completes the inductive argument and we have also shown that b_w is indeed the highest bid value. Hence we conclude that the protocol correctly computes the winning bid value for all $1 \leq j \leq l$.

5 Security against Rational adversaries

The security of our construction relies on the DDH assumption as well as security of the following building blocks: a) Security of the commitment scheme Com, b) Malicious security of the OT protocol Π_{OT} and c) Collision resistance of the hash function hash.

Our goal, informally, is to show that following properties hold:

1. Equilibrium (Part (1) of Definition 7). We first show that the dominant strategy is for parties to follow the protocol. This property relies on the security of our ABP encoding scheme (DDH), security of Pedersen commitments, collision resistance of the hash function as well as malicious security of the OT protocol .
2. Simulation (Part (2) of Definition 7). When parties follow the protocol, we show privacy using the simulation paradigm. Specifically, we show the existence of a simulator such that the view of the parties in the real world is indistinguishable from that output by the simulator. Here we use hiding property of Pedersen commitments, only semi-honest security of the OT protocol and indistinguishability of ABP encoding scheme (DDH).

5.1 Existence of Weakly Dominant Strategy Equilibrium (weak DSE)

We begin by setting some notation.

- b_i: Bid value of party P_i.
- v_i: Perceived private valuation of the auction item by P_i. Note that, motivation for P_i to participate in auction requires $v_i > b_i$.
- $u_i : \Gamma \times \{0,1\}^* \mapsto \mathbb{R}$ is the monetary utility function of party P_i mapping a strategy and all information on the BB to a value.
- $z_i : \Gamma \times \{0,1\}^* \mapsto \mathbb{R}$ is the information utility function of party P_i mapping a strategy and all information on the BB to a value.
- $U_i : \Gamma \times \{0,1\}^* \mapsto \mathbb{R}$ is the total utility function.
- $\mathcal{C}$: Set of parties caught deviating.

A party's utility function considers the gains or losses incurred by the party because of its actions. In addition to monetary considerations, parties are also curious to learn information about the bid values of other parties. For this, we consider a *information utility* function z_i. The total utility is a linear combination of the two components. Parties can have arbitrary privacy concerns (and value information differently). However, we assume certain restrictions on z_i, like not valuing "less" information higher than learning "more" information. These are reasonable, since it does not make sense to value learning, for instance, first bit of a certain party more than learning the first two bits of the same party. Similarly, parties do not value positively leaking their own information. Finally, we assume that parties value the monetary component of the utilities higher than the information component. That is, primarily, they want to win the auction; secondarily, they are concerned about revealing their information and learning information about other parties' inputs.

We now consider the monetary utility function of an individual party P_i from participating in Π:

$$u_i = \begin{cases} = v_i - b_i & \max(b_1, \ldots, b_n) = b_i \text{ and } P_i \text{ doesn't deviate} \\ = 0 & \max(b_1, \ldots, b_n) \neq b_i \text{ and } P_i \text{ doesn't deviate} \\ = -D/2 & P_i \text{ deviates and gets caught} \\ = -D & P_e \text{ identifies } P_i\text{'s cheating} \\ & \text{but } P_i \text{ rebuts and loses} \\ = D/2 & \text{if } i \neq e,\ P_e \text{ identifies cheater.} \\ & \text{But cheater does not rebut} \\ = \frac{D|\mathcal{C}|}{2(n-|\mathcal{C}|)} & P_i \text{ doesn't deviate, } \mathcal{C} \text{ is set of parties caught} \\ & \text{deviating} \end{cases}$$

In an ideal execution with a trusted party, each party P_i learns the winning bid value and the identity of the winning party. Let this information be valued at $Z_i \in \mathbb{R}^+$ by P_i. By correctness of our protocol, the information utility of the honest strategy of following the protocol is Z_i. We show that a deviating bidder never realizes more utility than when it is not deviating (Lemmas 3, 2 for monetary utility, and Lemma 4 for information utility).

Remark 1. We emphasize that an evaluator is also a bidder in the auction. Hence, all arguments in the security proofs below are applicable to evaluator as well. Wherever necessary, the role of evaluator and its impact on security are analyzed separately.

5.2 Strategies for Rational Parties

At the beginning of each round, a party can evaluate its utility based on the outcome of previous rounds and choose a strategy adaptively. During each round of the protocol, a party can choose either a 0-bit code, a 1-bit code or choose to abort based on its private random coins. A winning bidder not playing evaluator role has a deviating strategy $\mathsf{incorrectRebuttal}$. Additionally, if a party is also the evaluator (P_e), it can choose to compute the winning bid correctly or incorrectly. Since we are not considering any collusion among the parties, such a choice would be independent of choices made by other parties. As a result, we have following set of strategies available to the parties at the beginning of each round:

$$\Gamma_i = \{s_{0\to0}, s_{0\to1}, s_{1\to0}, s_{1\to1}, \mathsf{incorrectRebuttal}, \mathsf{abort}\}, \forall i \in [n] \setminus e$$

$$\Gamma_e = \{s_{0\to0}, s_{0\to1}, s_{1\to0}, s_{1\to1}, \mathsf{abort}, \mathsf{correctCompute}, \mathsf{incorrectCompute}, \mathsf{correctVerify}, \mathsf{incorrectVerify}\}$$

where (i)$s_{a\to b}$ denotes that the protocol requires bidder to use a bit a and bidder uses bit b. (ii)abort: The bidder stops participating in the protocol. (iii) correctCompute: The evaluator chooses to compute the winning bid value correctly. (iv) incorrectCompute: The evaluator chooses to compute the winning bid value incorrectly. (v) correctVerify: The evaluator chooses to verify winner's claim correctly. (vi) incorrectVerify: The evaluator chooses to verify winner's claim incorrectly. Among these $s_{0\to0}$, $s_{1\to1}$, correctCompute and correctVerify correspond to honest behavior or no deviation; and, $s_{0\to1}$, $s_{1\to0}$, incorrectRebuttal, incorrectCompute and incorrectVerify correspond to deviating strategies.

We can specify the strategies chosen by a party P_i for the entire auction as $\pi_i = (\pi_{i1}, \ldots, \pi_{il})$. For each round $j \in [l]$, P_i adaptively chooses its strategy based on the outcome of previous $j-1$ rounds and its bid value. Let π_{ij} denote the strategy used by P_i while following the protocol during jth round and π'_{ij} denote a deviating strategy. Then we have $\pi_{ij} \in \{s_{0\to0}, s_{1\to1}\}$ and $\pi'_{ij} \in \{s_{0\to1}, s_{1\to0}, \mathsf{incorrectRebuttal}, \mathsf{abort}\}$. Because of its special role, the evaluator P_e has two additional strategies available to it apart from the ones available to it as a bidder. Hence,

$$\pi_{ej} \in \{s_{0\to0}, s_{1\to1}, \mathsf{correctCompute}, \mathsf{correctVerify}\} \text{ and}$$
$$\pi'_{ej} \in \{s_{0\to1}, s_{1\to0}, \mathsf{incorrectCompute}, \mathsf{abort}, \mathsf{incorrectVerify}\}$$

For each party P_i, the choice of strategies by all other parties in the auction is denoted by π_{-i}. To emphasize the fact that these choices by other parties could be either honest or cheating strategies, we denote it by $\hat{\pi}_{-i}$. Every rational bidder's desire is to achieve their maximum utility. This means that, rational bidders will avoid any strategy that will push them out of the race. Once a party does drop out of the race too, it will avoid strategies which would further reduce its utility. (This, for example, is sure to happen if the party cheats and is detected.) In the following, we will show that honestly following the protocol is the *Weakly Dominant Strategy*. Utility realized as a result of any deviation is no better than what can be realized by following the protocol.

We want to establish that for a rational party, any deviation is irrational; i.e., certainly not utility enhancing and potentially utility diminishing. We will show this for the two deviations in next two Lemmas. Firstly observe that, if any bidder P_i aborts, the bidder gains nothing from its participation in auction. Moreover, the bidder ends up forfeiting its security deposit; thus having a net negative utility. Hence abort as a strategy is always utility decreasing and dominated by non-deviating strategies (either $s_{0\to0}$ or $s_{1\to1}$). Similarly the strategy incorrectRebuttal is not rational either. For this, consider a party P_i who claims win but is correctly caught cheating by the evaluator. If P_i chooses to rebut incorrectly, the opening of OT first message randomness by evaluator would certainly implicate P_i making him lose full deposit D. On the other hand, P_i can choose to accept the cheating and only lose deposit of $D/2$. Thus this strategy would not be adopted by any rational party.

We therefore need to focus our attention on only two strategies for a regular party P_i: $s_{1\to0}, s_{0\to1}$.

Proof while using Strategy $s_{1\to0}$ Consider an arbitrary bidder P_i who has followed the protocol till some arbitrary round j during the computation phase and is still in the race. Suppose that P_i has the bit $b_{ij} = 1$; i.e., P_i needs to use 1-bit code for computation during the jth round. Suppose P_i's strategy during the round j is $s_{1\to0}$ – i.e., uses a 0-bit code instead of 1-bit code during round j. For such a deviation we establish the following result.

Lemma 2. *For any PPT party P_i, let π_i be the honest strategy profile, and π'_i be the strategy profile in which P_i deviates using $s_{1\to0}$ in some round. Then, assuming that DDH assumption holds in $\mathbb{G}$, Com is a secure commitment scheme, and hash is a CRH, the strategy π_i weakly dominates the strategy π'_i for all $i \in [n]$ during all rounds as per Definition 4.*

Proof. Consider the round j where P_i deviates for the first time and uses the deviating strategy $s_{1\to0}$. Let $\hat{\pi}_{-i}$ denote strategies of parties other than P_i. Let us consider following cases on the winning bid at the end of the protocol

1. CASE: Winning Bid $= b_i$. It is easy to see that, regardless of his subsequent strategies, P_i will be detected as a cheating party and will have negative utility. If he chooses to not declare himself the winner, he will be caught during proof of not winning (because we assumed unique bids). If he declares himself the winner, he will get caught in his post-win verification protocol he runs with the evaluator. There's of course a chance that P_i is the evaluator i.e., P_e. In this case, any cheating is easily detected by as the evaluator will not be able to provide a proof of winning.

2. CASE: WINNING BID $\neq b_i$. Here we need to consider a few cases:
   (a) P_i was actually the highest bidder. In this case, P_i would have won the auction. It is easy to see that his utility in this case is strictly less than what it would have been if he was honest.
   (b) P_i was not the highest bidder and someone gets caught for cheating. In this case, either P_i also gets caught for cheating or he doesn't. In the former case, his utility has diminished and in the latter case, his utility is no more than what it would have been if he was honest.
   (c) P_i was not the highest bidder and someone else wins the auction. In this case, his utility is no more than what it would have been if he was honest.

   To argue that the protocol is a weakly dominant strategy, we need to show that there exists a strategy profile for other players, where P_i's utility from following the protocol is strictly more than not following it. To see this, consider the strategy profile $\hat{\pi}_{-i}$ wherein every party other than P_i uses the strategy $s_{1\to0}$ during every round. In this case case if P_i chooses to be honest, P_i is guaranteed to win. On the other hand, P_i's deviation would result in either losing the auction or getting caught to lose security deposit. Thus in this case, $u_i(\pi_i, \hat{\pi}_{-i}) > u_i(\pi_i', \hat{\pi}_{-i})$.

Remark 2. What this lemma demonstrates is that no party has monetary incentive to deviate using $s_{1\to0}$ as their first deviating strategy. Going forward, we will show this for all strategies and this will effectively mean that no party has a monetary incentive to deviate.

Proof while using Strategy $s_{0\to1}$ Now consider an arbitrary bidder P_i who deviates for the first time in round j and uses the strategy $s_{0\to1}$. We show that such a strategy is dominated by the honest strategy.

Lemma 3. *For any PPT party P_i, let π_i be the honest strategy profile, and π_i' be the strategy profile in which P_i deviates for the first time in round j and uses $s_{0\to1}$. Then, assuming that DDH assumption holds in $\mathbb{G}$,* Com *is a secure commitment scheme, and* hash *is a CRH, the strategy π_i weakly dominates the strategy π_i' as per Definition 4.*

Proof. Let $\hat{\pi}_{-i}$ denote strategies of parties other than P_i. Consider the round j where P_i has decided to use the deviating strategy $s_{0\to1}$ for the first time. We will show that irrespective of the strategic choices of other bidders, P_i cannot realize utility better than what it would have done without deviation.

The first observation is that whenever a party plays this strategy, he will forgo his chances of winning the auction. Even if his "fake" bid is the winning bid, he will never be able to prove his claim. Therefore, in this case, P_i will either get caught cheating or he will go under the radar, i.e., undetected. With the former, his utility diminishes compared to what it would have been if he had played honestly. With the latter, his utility is either the same as what it would have been if he had been honest (if his "true" bid was never the highest bid) or it diminishes (if his "true" bid was actually the highest bid).

To show that following the protocol is a weakly dominant strategy, we need to show that there exists a strategy profile for other players, where P_i's utility from following the protocol is strictly more than not following it. To see this, consider the strategy profile $\hat{\pi}_{-i}$ wherein every party other than P_i uses the strategy $s_{1\to0}$ during every round. In this case case if P_i chooses to be honest, P_i is guaranteed to win. On the other hand, P_i's deviation would result in either losing the auction or getting caught to lose security deposit. Thus in this case, $u_i(\pi_i, \hat{\pi}_{-i}) > u_i(\pi_i', \hat{\pi}_{-i})$.

Remark 3. To summarize, this lemma demonstrates that no party has monetary incentive to deviate using $s_{0\to1}$ as their first deviating strategy.

We now show that there is no monetary incentive to deviate for P_e.

Lemma 4. *For PPT evaluator P_e, let π_e be the honest strategy profile, and π_e' be the strategy profile in which P_e deviates in round j for the first time and uses* incorrectCompute *in some round during computation phase or using* incorrectVerify *during verification phase. Then, assuming that DDH assumption holds in $\mathbb{G}$,* Com *is a secure commitment scheme, Π_{OT} is a malicious secure OT implementation and* hash *is a CRH, the strategy π_e dominates the strategy π_e'. where $\hat{\pi}_{-i}$ denotes arbitrary strategies of parties other than P_i.*

$$u_e(\pi_e,\ \hat{\pi}_{-e}) \geq u_e(\pi_e',\ \hat{\pi}_{-e})\ \ \forall \hat{\pi}_{-e} \in \Gamma_{-e}$$

Proof. We first show that a rational evaluator P_e who deviates for the first time in round j and adopts the strategy π_{ej}' = incorrectCompute (to compute an incorrect winning bid value) could have used

correctCompute as the strategy and realized at least the same or better utility.

In our protocol, the evaluator P_e has a special role to play – that is to compute the winning bid value and also to verify winner's claim. At the end of the computation phase, suppose P_e declares the winning bid to be some value $b_r \neq b_w$ where b_w is the value of winning bid when the evaluator does not deviate. Note that b_w may be the result of some other parties deviating. Let us consider the following cases:

1. $\mathbf{b_w} > \mathbf{b_r}$: Then there exists party P_w such that there is at least some bit position $j \in [l]$ such that $b_{wj} = 1$ but $b_{rj} = 0$. Since P_e is expected to provide the 0-bit codes of each party whenever the computed output bit is 0, this means, P_e would have to compute the 0-bit code for P_w during round j. However, since P_e does not know the private key for P_w, this would be computationally infeasible for a PPT party P_e – assuming DDH assumption holds in the group $\mathbb{G}$. In such a case P_e would end up losing its deposit with overwhelming probability. On the other hand, if P_e had followed the protocol without deviation, then its utility would have been non-negative. Thus, $u_e(\pi_e, \hat{\pi}_{-e}) > u_e(\pi'_e, \hat{\pi}_{-e})$
2. $\mathbf{b_w} < \mathbf{b_r}$: This means, $b_r > b_i,\ \forall i \in [n]$. Suppose that b_w and b_r differ in jth bit. Consider a case where no party (other than the evaluator) deviates after round j. For all rounds after j, every bidder would be using a 0-bit code for computation. Thus the protocol would proceed in such a way that during the last decider round, all other parties would have used 0-bit code. Thus, every party would be able to produce a not-winning proof. As per the protocol, since all P_i $(i \neq e)$ have produced a not-winning proof, P_e stands implicated for wrong computation. P_e ends up losing its deposit.
   On the other hand if any other party cheats after jth round, that party would get caught and P_e gets paid. However, P_e would have anyway got paid even if P_e had followed the protocol without deviation. Hence, $u_e(\pi_e, \hat{\pi}_{-e}) \geq u_e(\pi'_e, \hat{\pi}_{-e})$.
   Thus, any strategic choice by P_e for incorrect computation of winning bid is dominated by the strategy to compute correctly.

Now let us consider the case when P_e uses strategy incorrectVerify during the verification phase. Recall that during verification phase P_e is engaged in an interaction with the winning party P_w to verify latter's claim. P_e starts by using the private randomness presented by the winning party P_w to compute the bit codes for each round $j \in [l]$. Then P_e uses the OT randomness shared by P_w to retrieve the bit codes actually used by party P_w during each round.

Now consider case when P_e claims that P_w's bit codes do not match. If P_e's claim is correct, P_w would be identified as cheating. On the other hand, if P_e's claim is incorrect, then P_w would refute the claim. In that case P_e would be forced to open its OT first message randomness and the choice bit β_{wj} corresponding to party P_w during round j. Recall that, during round j, P_e would have already written the OT first message for P_w to BB. Now if P_e is able to generate OT randomness such that it corresponds to the same OT first message, P_e can use the it to retrieve both messages of sender. This would violate security of OT. Since we are assuming a maliciously secure OT implementation, P_e can not succeed, except with negligible probability. As a result, P_e would get caught cheating and lose its deposit.
On the other hand, if P_e had not deviated, its utility would have been non-zero. Thus, $u_e(\pi_e, \hat{\pi}_{-e}) > u_e(\pi'_e, \hat{\pi}_{-e})$.

Remark 4. To summarize, this lemma demonstrates that the evaluator has no monetary incentive to deviate using incorrectCompute or incorrectVerify as her first deviating strategy. (We already demonstrated that the evaluator has no monetary incentive to deviate using $s_{0\to1}, s_{1\to0}$ as the first deviating strategy.)

Remark 5. We can argue that our protocol ensures fairness. Observe that evaluator can learn the output (i.e. the highest bid value) and decide not to write the same to BB – thus denying the output to other parties. However, such an action by evaluator would be considered as abort by the protocol and as a result evaluator would forfeit its deposit. Again, assuming a rational evaluator, fairness is ensured.

Remark 6. Combining the above lemmas, we have demonstrated that no party has a monetary incentive to deviate using strategies $s_{0\to1}, s_{1\to0}$, incorrectCompute and incorrectVerify as their first deviating strategy. They may offer information utility which we discuss next.

5.3 Information Utility

Recall that the information utility realized by P_i as a result of the protocol run without any deviation is $z_i(\pi_i, \pi_{-i}) = Z_i$. In the following, we show that this is the maximum information utility that any bidder can realize, unless the party P_i is willing to loose monetary utility.

Lemma 5. *For any PPT party P_i, let π_i be the honest strategy profile, and π_i' be the strategy profile in which P_i deviates using $s_{0\to1}$ or $s_{1\to0}$ in some round. Let $z_i(\pi_i', \hat{\pi}_{-i})$ be the information utility gained by P_i using the deviations. Then, assuming that DDH assumption holds in $\mathbb{G}$, Com is a secure commitment scheme, and hash is a CRH, the information utility realized by P_i is no more than what it would have realized without deviation; i.e.,*

$$z_i(\pi_i', \hat{\pi}_{-i}) \leq Z_i$$

(unless P_i diminishes his monetary utility), where $\hat{\pi}_{-i}$ denotes arbitrary strategies of parties other than P_i.

Here is a high-level idea of the proof. We make use of the fact that deviation using strategy $s_{1\to0}$ is not rational since party has to forego auction to learn the information. On the other hand, any deviation with $s_{0\to1}$ masks the bits used by other bidders. Thus we show that any party learns no more information as a result of deviation than it would have learnt by following the protocol. We also argue that the not-winning-proofs provided in a scenario resulting from adversarial action do not divulge any information about the bid values.

Proof. Suppose that the protocol had run through without any deviation. Let j' be last decider round in such run. In other words, round j' would be the last one in which the winner would have a 1-bit. Moreover, all rounds after j' would have every party using a 0-bit code. The information utility realized by a party P_i in such a case is, by definition Z_i.

Firstly we would like to consider the strategy $s_{1\to0}$. If any party P_i uses this strategy in some round j, then bit used by P_i would have no impact on the computation. As a result, the jth output bit would be same as the jth bit in winning bid value. Hence, P_i doesn't learn anything extra. However, in case P_i had the highest bid value, then P_i would end up learning jth bit of the second highest bidder. But for this, as shown in Lemma 2, P_i has to relinquish winning the auction. Since we have assumed that every rational party would value winning auction to be higher value than learning about others' bids, this strategy would not be considered by any rational party for learning information about other bid values.

Now we consider that P_i cheats between the rounds 1 and j' by using the strategy $s_{0\to1}$. For each such deviation, the round output would be a 1-bit independent of bits used by any other party, thus effectively masking inputs from all other parties. As a result, the information utility that can be realized by party P_i can not be more than Z_i.

If any party deviates beyond round j', the output would again be 1 for that round – and importantly, independent of bid values of any other party. This follows because for all rounds beyond j', every party uses 0-bit codes. Thus these deviations do not yield any extra information at all. Thus we conclude that the information utility z_i that a party P_i can realize from the computed output cannot be more than Z_i.

Now consider the winner of auction P_w who proves to the evaluator that it has used the correct bit codes during computation. If P_w has deviated during any round, the bit codes do not match and as a result the evaluator would identify P_w's cheating. But P_w might attempt to learn P_e's bits by rebuttal. However, as argued in section 5.2, this would incur monetary loss for P_w and hence not pursued.

We will also argue that, P_i does not gain any information from the proofs of not winning which are sought from all parties when there are no claimants for auction. This situation arises when the output is corrupted as a result of deviations and does not match any party's bid value.

Suppose that $j \neq j'$ is the last decider round, because of deviations. Since the output is corrupted because of deviations, no party claims the auction and everyone is asked to provide proofs of not-winning. For this, parties follow the protocol as described in Figure 1. Recall that, each party chooses its private key and OT random values independently, uniformly at random for each round. Hence, revealing these values for the round j as part of not-winning-proof does not divulge any information regarding the bits in position 1 to $j-1$ or $j+1$ to l (latter are all 0).

Moreover, as a proof of not-winning, each party proves that it had used a 0-bit code during the round j. But this does not reveal any information regarding the actual bit in position j for the party – since the 0-bit used during computation could be either the actual bit corresponding to bid value or it could be 0-bit because the party had fallen out of race. Thus the party P_i does not gain any extra information

from the proofs of not-winning shared for the last decider round j.

For the remaining rounds $j+1 \leq k \leq l$, the computed winning bits would be 0. A bidder could have used 0-bit for these rounds since it had a valid 0-bit for these positions in bid value. Or bidder might be using 0-bits since it has fallen out of race. Hence P_i would not learn anything extra.

Since we have considered an arbitrary bidder P_i, this holds for all $P_i, i \in [n]$. In summary, no PPT party P_i can realize any more information as a result of deviations than what it would have realized during an honest run:

$$z_i(\pi_i', \hat{\pi}) \leq Z_i$$

The equality holds in cases when the winner uses 1-bit code during the same rounds when P_i deviates. Thus the claim of lemma follows.

Putting Together Proof of Rational Security. Lemmas 2, 3, 4 and 5 show that no rational party will deviate from the protocol. Together, these establish Part (1) of Definition 7.

Theorem 2. *Let $P_i, i \in [n]$ be any PPT party participating in the first price auction, the protocol Π described in Figure 1. Then, assuming that DDH assumption holds in $\mathbb{G}$, Com is a secure commitment scheme, Π_{OT} is a malicious secure OT implementation and hash is a CRH, the protocol Π is a weakly dominant strategy equilibrium as per Definition 4.*

Proof. For party P_i, let π_i denote the strategy to follow the protocol, and π_i' denote any arbitrary efficient deviation. Let $\hat{\pi}_{-i}$ be arbitrary efficient strategies of other parties. From Lemmas 2, 3, and 4, we have $\forall \pi_i' \in \Gamma_i$,

$$u_i(\pi_i, \hat{\pi}_{-i}) \geq u_i(\pi_i', \hat{\pi}_{-i})$$

From Lemma 5 we have,

$$Z_i \geq z_i(\pi_i')$$

It follows that, for all P_i, $U_i(\pi_i, \hat{\pi}_{-i}) \geq U_i(\pi_i', \hat{\pi}_{-i}), \forall \pi_i' \in \Gamma_i$ and $\forall \hat{\pi}_{-i} \in \Gamma_{-i}$. We have that π_i is a weakly dominant strategy for P_i for all i. Thus, Π is a *Weakly Dominant Strategy Equilibrium.*

5.4 Privacy Enhanced weak DSE

Given the equilibrium from Theorem 2, to argue privacy, it suffices to show that nothing beyond the output is learned by parties who are non-deviating. We show security of our protocol against such semi-honest parties who follow the protocol but might attempt to learn more information from the protocol transcript, thus establishing Part (2) of Definition 7. We use the *ideal world – real world paradigm.*

We first present the ideal functionality, followed by the description of a simulator. We then show that the view of a semi-honest P_k in a real run of protocol is indistinguishable from the output of the simulator. We denote by $\mathsf{View}^{\Pi}_{P_k,\mathsf{real}}$, the distribution of the transcript of the protocol.

Ideal Functionality Let us denote the ideal functionality by $\mathcal{F}$ which operates with a set of parties $P = \{P_1, \ldots, P_n\}$ and an adversary Sim.

- Receives bids $b_i \in \{0,1\}^l$ from each party P_i
- Computes $b_w = \max(b_1, \ldots, b_n)$, and sends (P_w, b_w) to all parties and Sim.

Theorem 3. *Assuming Com is secure commitment scheme, DDH assumption holds in group $\mathbb{G}$, Π_{OT} is a semi-honest secure OT protocol and hash is a CRH, protocol Π specified in Figure 1 securely realizes the functionality $\mathcal{F}$ in the presence of semi-honest adversaries.*

Proof. We will construct a simulator for a PPT semi-honest party P_k, such that

$$\mathsf{View}^{\Pi}_{P_k,\mathsf{real}} \approx_c \mathsf{Sim}^{\Pi}_{P_k}$$

Simulator for evaluator ($k = e$). The simulator is invoked with: public parameters pp, bid value b_e of P_e and the output of ideal function, (P_w, b_w). Sim works as follows:

1. **Setup Phase**:

– Sim samples secret keys and randomness $x_{ij}, r_{ij} \leftarrow_R \mathbb{Z}_q$ on behalf of all parties $P_i, i \in [n]$. It samples OT sender randomness γ_{ij} and OT receiver randomness β_{ij} for all $i \neq e$ and $j \in [l]$. Computes the public keys $X_{ij} = g^{x_{ij}}, \forall i \in [n]$ and writes them to the BB. Sim also computes the hash H_i of concatenated string of private keys and OT randomness for each party.

– For each party $P_i, i \neq k, i \neq w$: Sim constructs commitments to bid values of 0, chooses $R_i \leftarrow_R \mathbb{Z}_q$, computes $c_i = g^{b_i} g_1^{H_i} h^{R_i}$ where $b_i = 0$ and writes the commitments to BB. For $i = w$, Sim chooses $R_w \leftarrow_R \mathbb{Z}_q$. Computes $c_w = g^{b_w} g_1^{H_w} h^{R_w}$. If $b_k \neq b_w$, Sim chooses $R_k \leftarrow_R \mathbb{Z}_q$. Computes $c_k = g^{b_k} g_1^{H_k} h^{R_k}$. These commitments are written to the BB.

2. **Computation Phase**: We consider following cases:

(a) $e = w$ (P_e is the winner): Sim generates 0-bit codes B_{ij} for all bidders $P_i, i \neq e$ for all rounds. For each round $j \in [l]$ and each party P_i, $i \neq e$, Sim samples β_{ij} and invokes $\mathsf{OT.R}_1(b_{wj}, \beta_{ij})$ to obtain otr^1_{ij} and writes them to BB. It also generates 0-bit codes B_{ej} whenever $b_{wj} = 0$. Using γ_{ij} sampled during setup phase and $M_{ij} \leftarrow_R \mathbb{G}$ it runs $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$ to obtain ots_{ij} for all parties $i \neq e$ and writes them to BB.

Sim writes the winning bits b_{wj} on BB for every round $j \in [l]$. Whenever $b_{wj} = 0$, 0-bit codes for all parties, $B_{ij}, i \in [n]$ are written to BB as P_e's proof of computation.

During verification phase, Sim writes randomness used for P_e's commitment R_e, all secret keys x_{ej}, r_{ej} and $\mathsf{OT.R}$ randomness β_{kj} for all $j \in [l]$, for all $k \in [n] \setminus \{e\}$ to BB.

(b) $e \neq w$ (P_e is not the winner): For party P_w, Sim generates bit codes B_{wj} for bits of winning bid b_w using the secret keys generated during the setup phase. For all other bidders, 0-bit codes B_{ij} are generated for all rounds. Sim runs the ABP between the bid values (b_w, b_e) to identify the bits d_{ej} that P_e uses during computation phase and generates 0-bit codes B_{ej} whenever $b_{wj} = 0$. For each round $j \in [l]$ and each party P_i, $i \neq e$, Sim samples β_{ij} and invokes $\mathsf{OT.R}_1(d_{ej}, \beta_{ij})$ to obtain otr^1_{ij} and writes them to BB. Sim uses γ_{ij} sampled during setup phase and $M_{ij} \leftarrow_R \mathbb{G}$ to run $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$ and obtains ots_{ij} for all parties $i \neq e$ and writes them to BB.

Sim writes the winning bits b_{wj} on BB for all $j \in [l]$. For all $j \in [l]$ where $b_{wj} = 0$, 0-bit codes for all parties are written to BB as P_e's proof of computation.

During verification phase, Sim writes randomness used for P_w's commitment, R_w, all secret keys x_{wj}, r_{wj} and $\mathsf{OT.S}$ randomness γ_{wj} for all $j \in [l]$ to BB.

In order to prove the indistinguishability, we consider the following hybrids:

– H_0: This is the real run of the protocol Π where real values for commitments and public keys are written to BB during setup phase. Real OT messages, proofs of computation are written to BB during computation phase, and real secret keys, $\mathsf{OT.S}$ randomness of the winner are written to BB during verification phase.

– H_1: This is same as H_0 except the following. During setup phase, all commitments are generated for bid values of 0, with random values of secret keys and $\mathsf{OT.S}$ randomness γ_{ij} – except for parties P_e and P_w. For these two parties, same values as in H_0 are retained. H_0 and H_1 are identically distributed since Pedersen Commitments are perfectly hiding (6).

– H_2: This is same as H_1 except for the following during computation phase rounds $j \in [l]$. If P_e is the winner, 0-bit codes B_{ij} are generated for all parties. $M_{ij} \leftarrow_R \mathbb{G}$ and $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$ is invoked to obtain ots_{ij} for each party and written to BB.

If P_e is not the winner: ABP is run between the bid values (b_w, b_e) to identify the bits d_{ej} that P_e uses during computation phase and 0-bit codes B_{ej} are generated whenever $b_{wj} = 0$. Bitcodes B_{wj} corresponding to winning bid b_w are computed, and for all other parties 0 bitcodes are generated. For all $i \neq e$, $M_{ij} \leftarrow_R \mathbb{G}$ and $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$ is invoked to obtain ots_{ij} and written to BB.

H_1 and H_2 are indistinguishable because of sender security of Π_{OT}.

– H_3: This is same as H_2 except for the following: During computation phase, $\forall j \in [l]$, if P_e is the winner, β_{ij} is sampled for all $i \neq e$, $\mathsf{OT.R}_1(b_{ej}, \beta_{ij})$ is invoked to obtain otr^1_{ij} and written to BB. If P_e is not the winner, β_{ij} is sampled for all $i \neq e$, $\mathsf{OT.R}_1(d_{ej}, \beta_{ij})$ is invoked to obtain otr^1_{ij} and written to BB.

H_2 and H_3 are indistinguishable because of receiver security of Π_{OT}.

Note that hybrid H_3 does not have any information about the bid values of parties other than P_e and that of P_w. Thus P_e cannot learn anything about losing bid values in H_3. In H_0, the view of P_e corresponds to the real run whereas H_3 corresponds to Sim's output. Moreover, by transitivity, $H_0 \approx_c H_3$.

Thus,

$$\mathsf{View}^{\Pi}_{P_k,\mathsf{real}} \approx_c \mathsf{Sim}^{\Pi}_{P_k}$$

Simulator for P_k ($k \neq e$). The simulator is invoked with: public parameters pp, bid value b_k of P_k, identity of P_e and the output of ideal function, (P_w, b_w). Sim works as follows:

1. **Setup Phase**: Sim in this phase is identical to the simulator for P_e described earlier.
2. **Computation Phase**:
   (a) $k = w$ (P_k is the winner): For P_k, Sim generates bit codes B_{kj} corresponding to winning bid b_k. For each round $j \in [l]$ and each party P_i $i \neq e$, Sim samples β_{ij} and invokes $\mathsf{OT.R}_1(0, \beta_{ij})$ to obtain otr^1_{ij} and writes them to BB. Using γ_{kj} sampled during setup phase and $M_{kj} \leftarrow_R \mathbb{Z}_q$, Sim runs $\mathsf{OT.S}(B_{kj}, M_{kj}, \gamma_{kj})$ to obtain ots_{kj} for P_k and writes to BB. For all parties $i \neq k$, 0-bit codes B_{ij} are computed and $M_{ij} \leftarrow_R \mathbb{Z}_q$. $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$ is invoked to obtain ots_{ij} and written to BB. Sim writes b_{wj} to BB for all $j \in [l]$. Whenever $b_{wj} = 0$, 0-bit codes for all parties are written to BB as proof of computation from evaluator .

   During verification phase Sim writes randomness used for P_k's commitment R_k, all secret keys x_{kj}, r_{kj} and $\mathsf{OT.S}$ randomness γ_{kj} for all $j \in [l]$ to BB.

   (b) $k \neq w$ (P_k is not the winner): Sim uses the ABP between (b_w, b_k) to identify the bits d_{kj} that P_k uses during computation phase and generates bit codes B_{kj} accordingly. For each round $j \in [l]$ and each party P_i $i \neq e$, Sim samples β_{ij} and invokes $\mathsf{OT.R}_1(0, \beta_{ij})$ to obtain otr^1_{ij} and writes them to BB. For all parties $i \neq k$, $i \neq w$, 0-bit codes are computed. For $i = w$, computes bit codes as per b_w. Using γ_{ij} sampled during setup phase and $M_{ij} \leftarrow_R \mathbb{Z}_q$ invoking $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$, Sim obtains ots_{ij} for all P_i and writes the same to BB. Sim writes b_{wj} on BB for all $j \in [l]$. In case $b_{wj} = 0$, 0-bit codes for all parties are written to BB as proof of computation from evaluator.

   During verification phase Sim writes randomness used for P_w's commitment R_w, all private keys x_{wj}, r_{wj} and $\mathsf{OT.S}$ randomness γ_{wj} for all $j \in [l]$ to BB.

In order to prove the indistinguishability we consider the following hybrids:

- H_0: This is the real run of the protocol Π where real values for commitments and public keys are written to BB during Setup Phase. Real OT messages, proofs of computation are written to BB during Computation Phase, and real secret keys, real $\mathsf{OT.S}$ randomness of the winner are written to BB during verification phase.
- H_1: This is same as H_0 except the following. During setup phase, all commitments are generated for bid values of 0, with random values of secret keys and $\mathsf{OT.S}$ randomness γ_{ij} – except for parties P_k and P_w. For these two parties, same values as in H_0 are retained. H_0 and H_1 are identically distributed since Pedersen Commitments are perfectly hiding (6).
- H_2: This is same as H_1 except for the following. During computation phase, if P_k is the winner, generate 0-bit codes for all senders except P_k. Generate P_k's bitcodes as per the winning bid b_k. Sample $M_{ij} \leftarrow_R \mathbb{G}$ and using γ_{ij} sampled during Setup Phase, invoke $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$ for all $i \neq e$, and write ots_{ij} to BB.

  If P_k is not the winner, for all $i \neq k, i \neq w$ generate 0-bit codes. ABP is run between the bid values (b_w, b_k) to identify the bits d_{kj} that P_k uses during computation phase. 0-bit codes B_{kj} are generated whenever $b_{wj} = 0$. For P_w, bitcodes as per the winning bid b_w are generated. Finally, for all i, sample $M_{ij} \leftarrow_R \mathbb{G}$ and invoke $\mathsf{OT.S}(B_{ij}, M_{ij}, \gamma_{ij})$, to obtain ots_{ij} which are written to BB. H_1 and H_2 are indistinguishable because of sender security of Π_{OT}.
- H_3: This is the same as H_2 except for the following: During computation phase, β_{ij} is sampled for all P_i, $\mathsf{OT.R}_1(0, \beta_{ij})$ is invoked to obtain otr^1_{ij} and written to BB.

  H_2 and H_3 are indistinguishable because of receiver security of Π_{OT}.

Observe that the hybrid H_3 does not have any information about the bid values of parties other than P_k and P_w. Thus P_k cannot learn anything about losing bid values in H_3. Also, in H_0, view of P_k corresponds to the real run whereas H_3 corresponds to Sim's output. Moreover, by transitivity, $H_0 \approx_c H_3$. Thus it follows that,

$$\mathsf{View}^{\Pi}_{P_k,\mathsf{real}} \approx_c \mathsf{Sim}^{\Pi}_{P_k}$$

The following theorem stating that Π is a *Privacy Enhanced Computational Dominant Strategy Equilibrium*, follows as a corollary of Theorems 2 and 3.

Theorem 4. *Let $P_i, i \in [n]$ be rational parties with respective utility functions $(U_1, \ldots, U_n)$ as described above. Assuming* Com *is a secure commitment scheme, DDH in group $\mathbb{G}$, Π_{OT} is a maliciously secure OT protocol and* hash *is a CRH, the protocol Π described in Figure 1 is a privacy enhanced computational weakly dominant strategy equilibrium as per Definition 7.*

6 Experimental Results

Our protocol was implemented in C++ with 1840 lines of code. We built upon OpenSSL and Boost open source libraries. The overwhelming cost of the protocol is computation over the group (even for 30 bidders, the total communication in the protocol was under 100 KB). Of this, exponentiation over Elliptic curve group using the secp256k1 curve forms bulk of the cost. In Table 2, we compare the number of exponentiations required by our protocol with other prior protocols such as [BHSR19,DGP22] (even though they suffer from non trivial leakage). Even so, we observe that our protocol makes 2X lesser exponentiation calls than prior works.

Protocol	**Communication Complexity**[a]	**Number of exponentiations**[b]
SEAL	$53nl\|\mathbb{G}\|$	$48nl + 44l$
FAST	$(9nl + 10n)\|\mathbb{G}\|$	$23nl + 20l + 8\log l + 2$
Protocol in [MNT09]	$O(n^3)$	NA [c]
Our Protocol	$(4nl + 7l + 5)\|\mathbb{G}\|$	$10nl + 5n - 9l$

Table 2: Comparison of efficiency of protocols.

[a] We have assumed that a group element from $\mathbb{G}$ is at least twice the size of element from $\mathbb{Z}_q$
[b] l is the number of bits in bid values, n is the number of parties participating in the auction.
[c] The work makes use of generic MPC protocols for auction.

We implemented our protocol and executed it on a single machine with Intel core i7 processor, 2.9 GHz. Figure 2 depicts the overall runtime of our protocol as a function of the number of bidders for different number of bits used to represent the bid values. This is plot of average of 3 runs for each number of bidders, with number of bidders ranging from $5 - 70$.

The run time for our protocol T is a function of number of group exponentiations $N = 10nl + 5n - 9l$. For a fixed number of bits l, the computation time $T \propto n$. Thus the run time scales linearly with the number of parties. This is corroborated by Figure 2.

Protocol [a]	**Data sent (in MB)**	**Run time (in sec)**
SEAL (has non-trivial leakage)	0.339	0.775
MP-SPDZ MASCOT (malicious)	995	0.81
MP-SPDZ MASCOT(semi-honest)	10.52	0.02
MP-SPDZ BMR (malicious)	8671	70.1
MP-SPDZ BMR (semi-honest)	1469	10.02
Our Protocol	0.03	0.141

Table 3: Comparison of implementation of protocols for $n = 10$ parties, $l = 10$ bits.

[a] The rows indicate the amount of data communicated and run time for computing the highest bid value.

The concrete efficiency of protocols is compared in Table 3. Among similar works, we only compare with SEAL implementation since a public implementation of FAST is not available. We have chosen the values of n and l to be same as the ones used in SEAL implementation – for better comparison and also as an optimum value for generic MPC protocols which do not scale well with number of parties. Note however that the choice of $l = 10$ need not be a limitation since the bid values can always be scaled by a constant factor. We have also used Multi-Protocol SPDZ (MP-SPDZ) [Kel20] library to benchmark MPC protocols that can evaluate the max value (i.e. identify the highest bid value) among a given set of 30

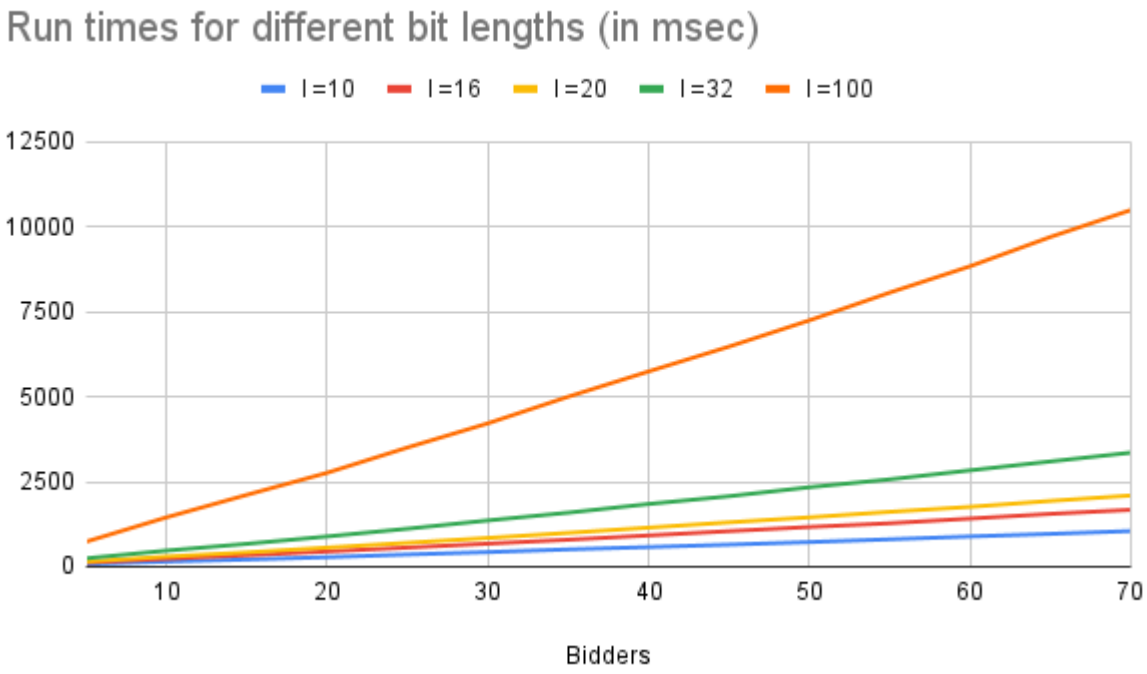


Fig. 2: Protocol Run time vs number of bidders

private inputs. The MP-SPDZ is an implementation of several MPC protocol variants, with a common high-level programming interface. Among these, we use MASCOT [KOS16] and BMR [LPSY19] protocols in malicious and semi-honest security modes for comparison. We have made our implementation available for public on https://github.com/girishabs/auction-protocol.

7 Conclusion

We construct a protocol for *First Price Auction* that is provably secure in the rational setting. Our implementation demonstrates that our construction is concretely efficient. We introduce the notion of *Privacy Enhanced Computational Weakly Dominant Strategy Equilibrium* as a solution concept, which we believe can be used to construct and analyse other cryptographic protocols. Our work leaves open several interesting questions about extending the protocol to other flavors of auctions such as Vickrey auctions, multi-unit auctions; and analysing other adversarial models like adaptive strategies, and colluding rational parties.

Acknowledgment

The research of the first author was supported by Core Research Grant CRG/ 2020/ 004488, SERB, Department of Science and Technology, India and a Google India Faculty Research Award. The research of the second author was supported, in part, by a Microsoft Collaborative Research Grant.

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.

AL07. Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 137–156. Springer, Heidelberg, February 2007.

AO12. Gilad Asharov and Claudio Orlandi. Calling out cheaters: Covert security with public verifiability. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 681–698. Springer, Heidelberg, December 2012.

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 G. Ray. SEAL: Sealed-bid auction without auctioneers. Cryptology ePrint Archive, Report 2019/1332, 2019. https://eprint.iacr.org/2019/1332.

BN00. Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, Heidelberg, August 2000.

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.

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.

DDM+20. Dominic Deuber, Nico Döttling, Bernardo Magri, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Minting mechanism for proof of stake blockchains. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part I*, volume 12146 of *LNCS*, pages 315–334. Springer, Heidelberg, October 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 *Annual International Cryptology Conference*, pages 112–130. Springer, 2000.

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.

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.

GMS08. Vipul Goyal, Payman Mohassel, and Adam Smith. Efficient two party and multi party computation against covert adversaries. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 289–306. Springer, Heidelberg, April 2008.

GOTZ21. Chaya Ganesh, Claudio Orlandi, Daniel Tschudi, and Aviv Zohar. Virtual asics: Generalized proof-of-stake mining in cryptocurrencies. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2021 International Workshops, DPM 2021 and CBT 2021, Darmstadt, Germany, October 8, 2021, Revised Selected Papers*, pages 173–191. Springer, 2021.

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.

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.

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.

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.

Lin13. Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 1–17. Springer, Heidelberg, August 2013.

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.

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.

Mor22. Ben Morrisroe. What will google's first price auction mean for publishers, 2022. https://www.publift.com/blog/what-will-googles-first-price-auction-mean-for-publishers.

MR12. Atsuko Miyaji and Mohammad Shahriar Rahman. Privacy-preserving set operations in the presence of rational parties. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 869–874. IEEE, 2012.

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

MT19. Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. Cryptology ePrint Archive, Report 2019/635, 2019. https://eprint.iacr.org/2019/635.

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

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.

SSS21. Peter Scholl, Mark Simkin, and Luisa Siniscalchi. Multiparty computation with covert security and public verifiability. *Cryptology ePrint Archive*, 2021.

A Oblivious Transfer Instantiation.

For our protocol, we use *Oblivious Transfer* (OT) for the interaction between the evaluator and other bidders. For the implementation, we have adopted the construction of *Oblivious Transfer* from [CSW20] which is described below. We consider two parties: sender S and receiver R for this.

1. **Public Input**: Group $\mathbb{G}$ with a generator g, group $\mathbb{Z}_q$, $h, T_1 \in \mathbb{G}$.
2. **R**:
   $T_2 \leftarrow_R \mathbb{G}$
   $\beta \leftarrow_R \mathbb{Z}_q$ and sets $G = g^\beta T_1^\alpha$ and $H = h^\beta T_2^\alpha$
   Send $(T_2, (G, H))$ to S.
3. **S**:
   Samples $s, t \leftarrow \mathbb{Z}_q$ and computes $z = g^s h^t$.
   Computes $C^{(0)} = G^s H^t \cdot M^{(0)}$ and $C^{(1)} = \left(\frac{G}{T_1}\right)^s \left(\frac{H}{T_2}\right)^t \cdot M^{(1)}$
   Send z and $(C^{(0)}, C^{(1)})$ to R.
4. **R**:
   R computes $M^{(\alpha)} = C^{(\alpha)} z^{-\beta}$

Fig. 3: Oblivious Transfer Protocol

B Contract Functionality

A contract is used for managing the security deposits in our protocol. Such a functionality can be realized as a contract within a legal framework using fiat currency or using Smart Contract using crypto currency.