Censorship-Resistant Sealed-Bid Auctions on Blockchains

Orestis Alpos*
Common Prefix
oralpos@gmail.com

Lioba Heimbach
Category Labs
liobaheimbach.cs@gmail.com

Kartik Nayak
Duke University
kartik@cs.duke.edu

Sarisht Wadhwa
Duke University
sarisht.wadhwa@duke.edu

***Abstract*—Traditional commit-and-reveal mechanisms have been used to realize sealed-bid on-chain auctions. However, these leak timing information, impose inefficient participation costs – the inclusion fee to be paid for adding the transaction on-chain – and also require multiple slots to execute the auction. Recent research investigates single-slot auctions; however, it requires a high threshold of honest parties.**

We present a protocol that addresses these issues. Our design combines timestamp-based certificates with censorship resistance through inclusion lists. The resulting protocol satisfies four properties, the first being a *strong hiding* property which consists of *Value Indistinguishability, Existential Obfuscation and User Obfuscation*. This not only ensures that the adversary cannot differentiate between two value of bids (as the previously defined *Hiding* property does in Pranav et al. [1]), but also that the very existence of a bid and the identity of the bidder remain obfuscated. The second property is *Short-Term Censorship Resistance*, ensuring that, if the underlying blockchain outputs a block, then the auction would contain bids from all honest users. The third is a new property we introduce, *Auction Participation Efficiency (APE)*, that measures how closely on-chain outcomes resemble classical auctions in terms of costs for participating users. And the fourth property is *No Free Bid Withdrawal*, which disallows committed bids from being withdrawn in case the bidder changes its mind.

Together, these properties yield a fair, private, and economically robust auction primitive that can be integrated into any blockchain to support secure and efficient auction execution.

1. Introduction

Auctions are fundamental to price discovery and resource allocation, and they have become increasingly important for blockchain systems. From NFT and token sales to block-space and network-resource allocation, blockchain platforms use a wide range of auction formats. Yet implementing auctions on a blockchain introduces unique challenges: blockchain transparency clashes with the confidentiality and strategic requirements of many auction types, and the irreversible, trustless nature of smart contracts demands careful mechanism design to ensure fairness and efficiency.

In a blockchain-based auction, bids are submitted as transactions to a smart contract that enforces the auction logic on chain. Each bid transaction must be included in a block to be recognized by the protocol, but only the consensus participant (proposer) selected for that slot decides which transactions are included. When bids are unencrypted, they appear in the mempool before inclusion, allowing anyone to observe them and enabling the proposer to include, delay, or replace them. This visibility means that the timing, presence, and identity of bidders can leak information about their valuations, making strong hiding guarantees essential in latency-sensitive auctions. Moreover, since all bid transactions are published on chain, even losing bids incur inclusion fees, which is not the case in traditional sealed-bid settings. Finally, many commit-and-reveal schemes allow bidders to freely abort or withhold their reveal transactions, enabling free bid withdrawal: a bidder can wait to see partial information from others and then selectively reveal only if doing so is profitable, undermining fairness and distorting incentives.

An effective blockchain sealed-bid auction protocol must therefore satisfy four key properties:

1) **inclusion fairness**, guaranteeing that all valid bids submitted before the deadline are included;
2) **withdrawal fairness**, ensuring that bidders cannot freely retract their bids after the deadline;
3) **hiding**, ensuring that bids remain confidential until they are collectively revealed;
4) **efficiency**, keeping participation costs comparable to classical off-chain auctions.

Achieving all four properties simultaneously is difficult on blockchains that rely on a single proposer per slot, since the proposer holds a temporary monopoly over transaction inclusion. A single proposer can selectively censor or delay certain bids, breaking inclusion fairness; the need for multi-phase or early commitments increases participation cost, undermining efficiency; and visibility of bids before inclusion compromises hiding. These challenges are especially acute in time-sensitive auctions where fairness depends on all bids submitted within the same short window being included together. Short-term censorship resistance is therefore critical: in settings such as liquidations or batch auctions in DeFi, a proposer who can delay or exclude competing bid transactions for even one slot can distort clearing prices or secure favorable outcomes. Thus, even correctly implemented commit–and-reveal or timed-commitment schemes can be undermined if a proposer withholds or reorders

honest bids at the decisive moment.

Recent efforts to mitigate the proposer's monopoly span a wide design spectrum. Some aim to remove the single proposer monopoly entirely by restructuring consensus itself [1], offering strong theoretical guarantees but at the cost of deep protocol changes. Others take a lighter approach, introducing inclusion [2], [3] or commit–and-reveal [4]–[6] mechanisms within existing architectures to improve fairness in practice. Yet these remain incomplete: they lack strong hiding and cannot fully guarantee inclusion in narrow, time-bounded participation windows. The result is a persistent trade-off between robustness and practicality.

In this work, we introduce a lightweight, consensus-adjacent mechanism that enables fair and efficient participation in time-sensitive blockchain applications. Our protocol combines timestamp-based certificates, bid hiding, and fork-choice enforcement to ensure inclusion fairness and confidentiality. Timestamp certificates provide verifiable evidence of when each bid became visible to the network, while the hiding mechanism conceals both the content and the existence of bids until the designated release time, preventing premature information leakage. Finally, by guaranteeing that any honest bid that reaches the network before the auction deadline will be taken into account, the protocol provides strong *short-term censorship resistance*. Importantly, only the winning bid is ever revealed on-chain, yielding *auction participation efficiency (APE)*, as losing bidders incur no cost. Together, these properties yield a fair, private, and economically efficient auction primitive that can be deployed within existing blockchains with minimal modification.

Our Contributions. We present a censorship-resistant sealed-bid auction protocol that operates directly on blockchains with minimal modifications to the consensus.

A key contribution of this work is to systematically identify and formalize the security properties that sealed-bid auctions must satisfy in latency-sensitive blockchain environments. We outline why prior definitions of hiding and censorship resistance fall short in these settings and introduce strengthened variants tailored. Our protocol is designed to satisfy these stronger properties while remaining lightweight.

1) *Selective Short-Term (SST) Censorship Resistance.* Our protocol achieves an enhanced SST censorship resistance, ensuring that any honest bid visible to the network before the auction deadline is considered.
2) *(Relaxed) t_h-Strong Hiding.* We show that hiding alone is insufficient in time-sensitive auctions and formalize a stronger requirement that captures both existence leakage and user identity leakage, namely *existential indistinguishability* and *user anonymity*. While our protocol does not achieve these full guarantees, it satisfies a relaxed form of t_h-strong hiding – combining indistinguishability, existential obfuscation, and user anonymity among active users – which together hide bid contents and obfuscate their association to users or specific auctions before the auction deadline.
3) *No Free Bid Withdrawal.* We provide an additional fairness guarantee by preventing bidders from selectively aborting or withdrawing bids after seeing partial information; deviations after the hiding period incur a cost.
4) *Auction Participation Efficiency (APE).* We introduce a participation-efficiency property for blockchain auctions and demonstrate that our protocol achieves it: honest losing bidders pay nothing, since only the winning bid is settled on chain.

To the best of our knowledge, our protocol is the first to achieve this combination of properties while still being practically deployable and imposing minimal on-chain overhead.

2. Model and Notation

Our goal is to design a protocol that enables traditional sealed-bid auctions to be conducted on-chain while ensuring that every bid is included during the auction.

2.1. Notation

In pseudocode, the keyword **function** denotes a method of a smart contract, i.e., a method that is executed *on-chain*, while keyword **offline** denotes a block of code executed by users *off-chain*. When referring to zero-knowledge proofs, we denote by $\text{ZKPROVE}(\mathsf{statement}, \mathsf{witness})$ a function that returns a proof π that $\mathsf{statement}$ is true with respect to $\mathsf{witness}$, and by $\text{ZKVERIFY}(\mathsf{statement}, \pi)$ a function that verifies that π is a valid proof for $\mathsf{statement}$.

2.2. Network and Participants

Network Model. We assume a *synchronous network* where any message sent by an honest party is guaranteed to be received by all other honest parties within a known maximum delay Δ. This also means that we assume a $\Delta-$synchronized clock for all parties, i.e., the clocks they maintain are within Δ of each other.

Communication model. Parts of our protocol use a *gossip network*. The goal is to obfuscate the origin of the message, hence a message sent through this network does not contain user identifiers. Similarly, a reply to the message is sent through the gossip network with some known identifier to indicate the original message. If a message is sent through the gossip channel, we assume that the message is received by all parties within Δ_g delay.

System Participants. Our system consists of three main types of participants:

- **Users ($\mathcal{U}$):** A set of entities that wish to participate in an auction. A user $u \in \mathcal{U}$ creates a transaction tx, which includes its contents (i.e., a bid) and a fee f. A user uses the gossip network to interact with the timestamping replicas. To represent the value of bid contained in the transaction tx, we use the notation $\text{val}(tx)$

- **Timestamping Replicas ($\mathcal{P}_{ts}$):** A permissioned set of $n_{ts} = 2f_{ts} + 1$ replicas, responsible for providing a timestamp for any transaction they receive. On receiving a transaction (or a blinded transaction, e.g., its hash), a timestamping replica $P_{ts,} \in \mathcal{P}_{ts}$ signs the transaction after attaching the current local clock time. We assume an honest replica timestamps every transaction it receives, and that the adversary controls at most f_{ts} Timestamping Replicas.
- **Inclusion List (IL) Proposers ($\mathcal{P}_{il}$):** A permissioned committee of n_{il} parties, responsible for censorship resistance. Each inclusion list proposer $P_{il} \in \mathcal{P}_{il}$ contributes to constructing the inclusion list by choosing transactions from the mempool. We assume the adversary controls at most f_{il} of the IL Proposers. The exact value for f_{il} depends on the inclusion-list mechanism used. In this paper, we will be using FOCIL [2], hence we assume existential honesty in the set $\mathcal{P}_{il}$, i.e., $f_{il} = |\mathcal{P}_{il}| - 1$.

The sets $\mathcal{P}_{ts}$, $\mathcal{P}_{il}$, and $\mathcal{U}$ can overlap, but for simplicity we assume in our analysis that the sets are disjoint. For all of these sets, we call *corrupted* a party controlled by the adversary – in which case the party can behave arbitrarily – and *honest* otherwise.

2.3. Desired Properties

A secure sealed-bid auction protocol in our model must satisfy the following four core properties:

Selective Short-Term Censorship Resistance. The first property we require is censorship resistance. Fox et. al. [7] show the importance of censorship resistance for on-chain auctions. Without censorship resistance, a malicious bidder can always ensure that honest bidders have lower utility than in classical auctions.

In blockchain systems, censorship resistance has traditionally been defined as a form of liveness: an honest user's transaction is guaranteed to be included eventually, assuming blocks continue to be produced. For sealed-bid auctions, however, eventual inclusion is insufficient. Bids must be included *before* the auction deadline t_{end} for the bidder to participate. Even a short delay, e.g., a malicious proposer withholding an honest bid and instead including its own lower bid just before the deadline, can prevent the honest bid from being included in the auction at all.

While prior work has introduced short-term notions of censorship resistance, these guarantees apply only *conditional on the block for that slot being produced* [1]. If the adversary controls the final proposer before the auction deadline t_{end} and knows that no subsequent block will be built, it can prevent timely honest bids from being included simply by not producing the final block, while still including its own bid in the preceding block. Auctions therefore require a form of short-term censorship resistance that guarantees inclusion in the auction's input set before t_{end}, independent of whether a particular block is produced.

Thus, we introduced a refined definition of *SST Censorship Resistance* that ensure that every bid submitted by an honest user before the auction deadline is guaranteed to be included in the auction's input set.

Definition 1 (Selective Short-Term Censorship Resistance)**.** *A sealed-bid auction protocol satisfies* Selective Short-Term (SST) Censorship Resistance *if, for every auction and every honest bidder u, if u's bid tx is sent to the protocol before the auction deadline t_{end}, then tx is included in the auction's input bid set used to determine the outcome.*

Ideally, we would want to achieve the above, but given network delays, it is very hard to achieve. Thus, we define the following relaxation,

Definition 2 (δ-SST Censorship Resistance)**.** *A sealed-bid auction protocol satisfies* δ-Selective Short-Term Censorship Resistance *if, for every auction and every honest bidder u, the following holds: if u's bid tx is sent to the protocol at least time δ before the auction deadline t_{end}, then tx is included in the input bid set used to determine the outcome.*

t_h-Strong Hiding. Second, auctions require a *Hiding* property. Existing definitions of hiding in sealed-bid auctions typically guarantee only value indistinguishability: an adversary cannot distinguish between two bids of different values [1], [5]. However, this definition is incomplete for latency-sensitive auctions such as MEV auctions.

First, the timing of the bid creation leaks information. In an auction where prices fluctuate, the time at which a bid is generated can be strongly correlated with its value [1]. If an adversary knows when a transaction was generated, and all parties know the optimal bid at that moment (e.g., the instantaneous arbitrage value between on-chain and off-chain markets), then the adversary can infer the bidder's value even if the bid is cryptographically hidden.

Second, even the existence of a bid from a particular user can leak strategic information. If an adversary can tell that no other bids have been submitted, it can safely underbid. Similarly, if the adversary can identify the party that submitted the bid, it can compute the expected bid amount based on the prior that it has on the user. This is acceptable in second-price auctions but harmful in first-price auctions and MEV settings. Thus, it is not enough to hide the value of the bid; one must obfuscate whether a bid was submitted at all, and completely hide the identity of the user that submitted the bid.

Informally, we require not only that the bid's value is hidden, but also that the very existence of the bid is obscured. Thus, we define a more refined t_h*-Strong Hiding* property below.

Definition 3 (t_h-Strong Hiding)**.** *Let* $\mathsf{negl}(\cdot)$ *denote a negligible function and let* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{hiding}}(\lambda)$ *denote the advantage of a PPT adversary $\mathcal{A}$ in a game, parameterized by the security parameter λ. Further, for an auction with submission deadline t_{end}, let $t_h > t_{\text{end}}$ denote a point in time. A protocol Π satisfies t_h-*Strong Hiding *if for every PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{hiding}}(\lambda) \leq \mathsf{negl}(\lambda).$$

The advantage is defined with respect to the following three indistinguishability games:

(i) ***Indistinguishability.*** *The adversary $\mathcal{A}$ outputs two transactions (tx_0, tx_1). A challenger samples $b \xleftarrow{\$} \{0,1\}$ and submits tx_b to the protocol. At time t_h, the adversary outputs a guess b'. The advantage is*

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{A}}(\lambda) := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

(ii) ***Existential Indistinguishability.*** *The adversary $\mathcal{A}$ outputs a transaction tx. A challenger samples $b \xleftarrow{\$} \{0,1\}$; if $b = 0$ the challenger submits tx to the protocol, and if $b = 1$ the challenger does not submit tx to the protocol. At time t_h, the adversary outputs a guess b'. The advantage is*

$$\mathsf{Adv}^{\mathsf{exist}}_{\mathcal{A}}(\lambda) := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

(iii) ***User Anonymity.*** *The adversary $\mathcal{A}$ outputs a transaction tx and two user identifiers (u_0, u_1). A challenger samples $b \xleftarrow{\$} \{0,1\}$; if $b = 0$ the challenger submits tx to the protocol on behalf of user u_b. At time t_h, the adversary outputs a guess b'. The advantage is*

$$\mathsf{Adv}^{\mathsf{user}}_{\mathcal{A}}(\lambda) := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

Finally, we define

$$\mathsf{Adv}^{\mathsf{hiding}}_{\mathcal{A}}(\lambda) := \max\left\{\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{A}}(\lambda),\, \mathsf{Adv}^{\mathsf{exist}}_{\mathcal{A}}(\lambda),\, \mathsf{Adv}^{\mathsf{user}}_{\mathcal{A}}(\lambda)\right\}.$$

Note that in a protocol where the hiding time t_h and the submission deadline t_{end} satisfy $t_h > t_{\text{end}}$ (such as the protocol presented in this paper), confidentiality holds strictly beyond this deadline. Consequently, the adversary cannot learn the content of any bid, and the very existence of each bid remains obfuscated until no new bids for that auction can be submitted.

Looking ahead, our protocol will not achieve the full t_h-*Strong Hiding* property. However, we keep these definitions as ideal properties for an auction protocol and pose the open question of whether t_h-*Strong Hiding* can be satisfied. Our protocol, will provide *indistinguishability*, but it does not satisfy *existential indistinguishability* and *user anonymity*. Instead, it achieves what we refer to as *existential obfuscation* and *user obfuscation*. Informally, although an adversary may observe that messages are being submitted to the system, it cannot determine which of the parallel auctions they belong to. As a result, the existence of a bid in any specific auction is effectively obfuscated.

Definition 4 (Existential Obfuscation within $\textsc{AucSet}(\tau)$)**.** *The adversary $\mathcal{A}$ outputs a transaction tx and two Auction IDs AucID_0 and AucID_1, where $\mathsf{AucID}_0, \mathsf{AucID}_1 \in \textsc{AucSet}$ (which is the set of auctions active at timestamp τ). A challenger samples $b \xleftarrow{\$} \{0,1\}$; the challenger submits tx for the auction AucID_b. At time t_h, the adversary outputs a guess b'. The advantage is*

$$\mathsf{Adv}^{\mathsf{exist}}_{\mathcal{A}}(\lambda) := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

Comparing this to Existential Indistinguishability, this implies that if the adversary observes a transaction tx at timestamp τ, then the probability that this transaction is for Auction AucID is $\frac{1}{|\textsc{AucSet}|}$

Although existential obfuscation is weaker than full existential indistinguishability, it meaningfully mitigates the key sources of leakage identified above. By preventing the adversary from linking a bid to specific auctions, it disrupts timing-based inference of bid values that could otherwise be exploited strategically. While this does not eliminate all possible forms of existence leakage, it is a substantial step toward the privacy needed in latency-sensitive auctions, even if it falls short of the full t_h-*Strong Hiding* guarantee.

Finally, we note that our protocol achieves user anonymity only among the set of registered users, i.e., users with active deposits (see Section 4.2); we refer to this as *user obfuscation among active users*.

Definition 5 (User Obfuscation within $\mathcal{U}$)**.** *The adversary $\mathcal{A}$ outputs a transaction tx and two user identifiers $(u_0, u_1 \in \mathcal{U})$. A challenger samples $b \xleftarrow{\$} \{0,1\}$; if $b = 0$ the challenger submits tx to the protocol on behalf of user u_b. At time t_h, the adversary outputs a guess b'. The advantage is*

$$\mathsf{Adv}^{\mathsf{user}}_{\mathcal{A}}(\lambda) := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

Together, *indistinguishability*, *existential obfuscation*, and *user obfuscation among active users* form what we call a *Relaxed* t_h-*Strong Hiding* property, which mitigates the key timing and identity leakage channels.

No Free Bid Withdrawal. From the *Hiding* property we have that, until time t_h, the adversary has no information about the bids that have been submitted. However, after time t_h, the protocol may allow bids to be revealed. Of course, for the *Hiding* property to make sense, the protocol will not allow new bids to be created after this point. However, this is not sufficient – if the adversary has the ability to *cancel* bids, i.e., to hide their existence or remove them from the protocol, then this would be effectively the same as being able to create bids after t_h. For example, an adversary could submit many bids and suppress all but the one that yields the most favorable outcome.

Such manipulations arise broadly in commit-and-reveal schemes, where the ability to decide whether a bid becomes visible after observing partial information from others enables strategic withholding or selective abort. Hence, we require a *No Free Bid Withdrawal* property, complementing *Hiding*, which ensures that once a bid is submitted, it cannot be withdrawn after the auction deadline without incurring a prohibitive cost. We formalize this property next.

Definition 6 (No Free Bid Withdrawal)**.** *A sealed-bid auction protocol satisfies the* No Free Bid Withdrawal *property if the following holds. Consider an execution of the protocol in which all users are honest after time t_h and the winning bid is the bid contained in transaction tx. Now consider a second execution that allows adversarial users after time t_h, resulting in a winning bid contained in transaction tx'. If the bid b' in tx' is strictly smaller than the bid b in tx, i.e.,*

$b' < b$, then the user submitting tx must incur a cost for doing so.

The intuition is that, if an execution e exists where transaction tx' contains the winner bid b' and $b' < b$, whereas the honest execution returns b as the winner bid, then the user that submitted bid b has effectively withdrawn b in execution e. In other words, the adversary is not able to change the winner bid after time t_h to a lower bid b' without incurring a cost.

Auction Participation Efficiency. We introduce a fourth property – *Auction Participation Efficiency* – which captures how closely an on-chain auction emulates the payoff dynamics of its classical counterparts. In traditional auctions, only the winning bidder(s) pay their bid amounts, while all other participants incur no cost for participation. In contrast, on-chain implementations require every bidder to submit a transaction containing their bid, each of which must pay an inclusion fee regardless of the auction outcome. This leads to a situation in which most parties end up with a negative utility at the end of auction, their expected value is inflated from when they win the auction.

Definition 7 (Auction Participation Efficiency)**.** *An on-chain auction protocol satisfies* Auction Participation Efficiency (APE) *if there is no participation cost incurred by the losing bidders.*

3. Background

3.1. Overview of the FOCIL Protocol

FOCIL (Fork-choice enforced Inclusion Lists) [8] has been proposed as a mechanism to improve transaction-inclusion guarantees on Ethereum by imposing constraints on Ethereum block builders. An instance of the FOCIL protocol is run for each Ethereum slot, which, according the Ethereum consensus protocol, has a duration of 12 seconds. Each instance involves a set of *IL Proposers*, a set of *Attesters*, and a set of *Validators*, once of which is the *block builder*. An instance works as follows.

A set of IL proposers is selected from the validator set. Each IL proposer monitors the public mempool and constructs an IL of Ethereum transactions. Each IL can be at most 8,KB, and proposers are free to choose their own construction policy – for example, prioritizing transactions by highest priority fee. Once built, the IL is gossiped to validators and to the block builder for the current slot. IL proposers may construct ILs until the 8th second of the slot.

Once a validator receives an IL, it verifies certain conditions. Among others, it verifies that (1) the size of the IL does not exceed the maximum allowed size, (2) it is an IL created for the current slot, (3) it is signed by an IL proposer of the current slot and (4) that IL proposer has not sent a different IL for the current slot. If verification succeeds, the validator further propagates the IL. Validators process and propagate ILs until the 9th second of the slot.

The block proposer (which is chosen from the set of validators) collects ILs up to the 11th second of the slot. At that point, it builds a block consisting of the transactions in *all* valid ILs it has received – except for transactions that do not fit in the block, which the block proposer may choose to omit. The block is propagated to the rest of the validators.

A set of IL attesters is also chosen from the set of all validators. An attester receives the block for the current slot and only votes for it if it does not omit any transaction from an IL locally seen (up to the 9th second of the slot) – except for transactions that do not fit in the block. A block not signed by sufficiently many attesters is not considered canonical and cannot be part of the canonical chain.

Looking forward, we will use FOCIL as the second sub-protocol in our construction, but with some modifications what we introduce later.

3.2. Blockchain Protocol

We assume a standard single-leader proof-of-stake (PoS) blockchain with a deterministic slot schedule that supports smart contracts (e.g., Ethereum). The deterministic slot schedule means that slots occur at precise, pre-defined intervals known to all participants, allowing the network to remain synchronized in time. In each slot, a validator acts as the *leader* to collect transactions, execute them, and propose a new block, while other validators attest to finalize it under the protocol's fork-choice rule. This fork-choice incorporates validity conditions, that are specifically defined in Section 4.4, paragraph on FOCIL in our construction.

This base protocol ensures consensus on both the order of transactions and the resulting on-chain state. In our construction, we only modify it to require that the inclusion of winning auction bids is enforced through a fork-choice–based mechanism, as described in Section 4.4.

4. Protocol Description

Before presenting the details, let us see the protocol as an abstraction and present the two types of participants that interact with it.

The first is the users $\mathcal{U}$ that wish to participate in an auction by submitting a bid. Each user registers in the pre-phase by locking a deposit and generating a private commitment that will later be used to prove eligibility. The auction protocol will require a user $u \in \mathcal{U}$ to act twice, i.e., to send a message, receive responses from the protocol, and then send a second message during the bidding phase.

The second is an auctioneer a (as defined in the underlying PoS protocol), that will obtain the bids from our protocol. An auctioneer registers its auction during the preparatory phase, specifying parameters such as the auction identifier, start and end times, and the item being auctioned.

Our protocol enables censorship-resistant sealed-bid auctions. It combines two sub protocols: a timestamping network that provides verifiable bid creation times, and a censorship-resistant inclusion mechanism based on ILs. Together, they achieve the properties presented in Section 2.3.

Figure 1 gives an overview of all function calls made during the auction. At a high level, the protocol operates as follows.

Pre-Phase A (Auction Registration). The auctioneer registers the auction AucID in the global contract $\mathcal{DC}$. During this process, the auctioneer specifies the parameters and reserves settlement capacity on-chain. This ensures unique and verifiable auctions and limits concurrent auctions to guarantee sufficient on-chain settlement capacity.

Pre-Phase B (User Deposit and Membership Proof). Each user u registers by locking a deposit D in the global contract $\mathcal{DC}$ and committing to a private secret ρ_u, forming $C_u = \textsc{Hash}(\rho_u)$ that is stored in a public Merkle-tree registry. The deposit mechanism will be used to ensure that bidders cannot submit multiple bids and selectively reveal only the most favorable one. Each deposit will remain locked until the user, for each auction it has participated in, either reveals its bid or proves that the bid was smaller than the winning bid. This will assist in achieving the *No Free Bid Withdrawal* property.

We remark that a user u registers only once with a fixed deposit D. The deposit remains active until the user withdraws it and enables u to participate in all auctions under the global contract $\mathcal{DC}$, as long as it follows the protocol in each auction.

Phase 1 (Timestamping). For every auction AucID, the user proves its eligibility to participate by deriving a pseudonymous handle $h_u = \textsc{Hash}(\rho_u \parallel \mathsf{AucID})$ and generating a zero-knowledge *proof of membership*, showing that h_u corresponds (through the secret ρ_u) to an active deposit. An eligible user obtains a timestamp for a commitment of its bid by interacting with the timestamping replicas $\mathcal{P}_{ts}$.

A timestamper only signs a commitment if the user provides a handle h_u and a verifying proof. Collectively, $\mathcal{P}_{ts}$ certify the creation time of the bid without learning its content, producing a *timestamp certificate* as public evidence of bid existence prior to the auction deadline. This phase will assist in achieving the *t-Simultaneous Release* property, ensuring that all bids can be revealed simultaneously at release time while remaining hidden beforehand.

Phase 2 (Bid Submission and Inclusion). Each user now reveals its bid by submitting it, together with the timestamp certificate, to the inclusion list proposers $\mathcal{P}_{il}$, which ensure that all valid bids reaching the network before the auction deadline are eligible for inclusion. By enforcing inclusion through fork-choice rules, the protocol prevents censorship and guarantees that every timely bid from an honest user is considered in the auction, while only the winning bid is ultimately settled on-chain. This phase will assist in achieving the *SST censorship resistance* property.

4.1. Pre-Phase A: Auction Registration

Before any user deposits or bids can occur, each auctioneer must register by deploying and initializing an on-chain deposit contract. This pre-phase ensures that all subsequent auctions are uniquely identifiable, at most n_A-overlapping, i.e., our protocol supports n_A parallel auctions.

Each auctioneer a registers an auction in a global contract $\mathcal{DC}$ that maintains the registry of deposits and auction parameters. In Algorithm 1 we show the algorithm exposed by $\mathcal{DC}$ for this purpose. Each registered auction is identified by a unique and strictly increasing *Auction ID* (AucID). The protocol supports at most n_A auctions ongoing at the same time. For each auction, the auctioneer specifies

$$(\mathsf{AucID}, \mathsf{item}, t_{\text{start}}, t_{\text{end}}),$$

where item denotes the asset being auctioned, and $t_{\text{start}} < t_{\text{end}}$ define the auction's active window.

We assume a proof-of-stake blockchain protocol with a deterministic slot schedule. A small fraction of blockspace in each slot is reserved for auction-settlement transactions. This reserved blockspace unconditionally includes the winning bids on chain; since only a single winning bid must be posted per auction (see Section 4.4) and at most n_A auctions run in parallel, the total reserved capacity required per slot is small. If one or more preceding slots fail to produce a block, the protocol automatically increases the reserved capacity in the next available slot to ensure that all pending auction settlements can be included. We discuss this approach further in Section 6.

Algorithm 1 Auctioneer Registration and Auction Setup

Require: Auctioneer identifier a
Ensure: Register auctions, at most n_A-overlap

On-chain functions on contract $\mathcal{DC}$:

global ctr $\leftarrow 0$
global auctionsPerSlot$[i] \leftarrow 0, \forall i \in \mathbb{N}$
global auctions $\leftarrow \emptyset$ ▷ map from AucID to auction parameters
function REGISTERAUCTION(a, item, $t_{\text{start}}, t_{\text{end}}$)
  for $t \in [t_{\text{start}}, t_{\text{end}}]$ **do** ▷ max n_A auctions per slot
    requireauctionsPerSlot$[t] < n_A$
  for $t \in [t_{\text{start}}, t_{\text{end}}]$ **do**
    auctionsPerSlot$[t] \leftarrow$ auctionsPerSlot$[t] + 1$
  AucID $\leftarrow$ ctr
  auctions[AucID] $\leftarrow$ (item, $t_{\text{start}}, t_{\text{end}}, S$)
  ctr $\leftarrow$ ctr $+ 1$
function GETAUCTIONRECORD(AucID):
  return auctions[AucID]

Note that with Pre-Phase A, we ensure the uniqueness and public verifiability of auction results, as all participants agree on the existence and the parameters of each auction. We also ensure that the number of concurrent auctions remains limited, so that sufficient capacity exists to unconditionally enforce the inclusion of bids. This foundation enables subsequent phases to rely on consistent auction timelines and prevents equivocation by auctioneers. In practice, to prevent the auction contract from being dominated by low-value auctions, one could introduce a permissioned

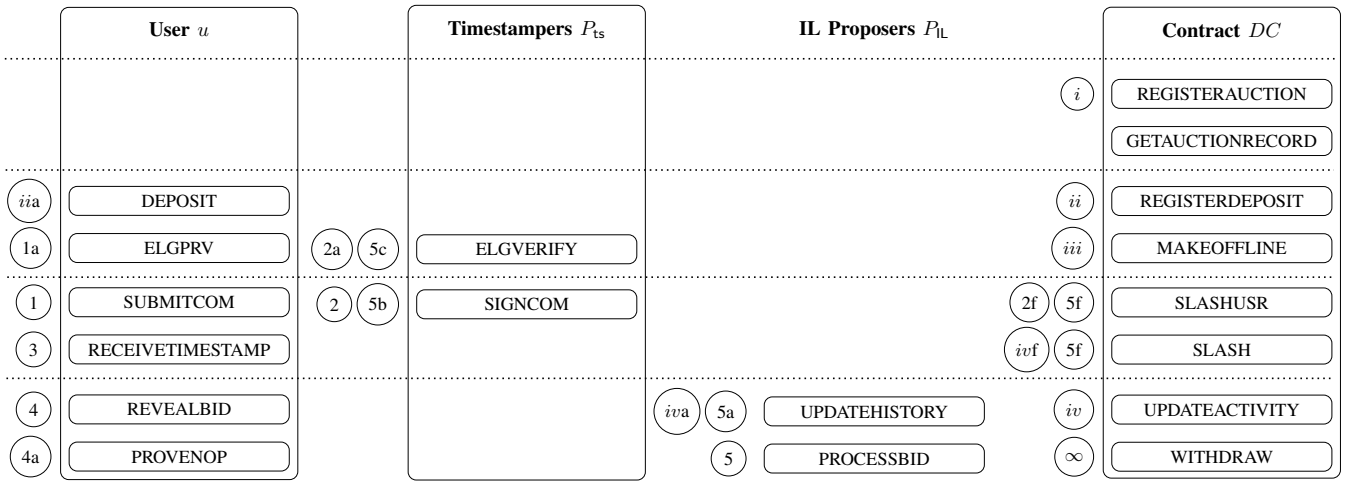

Figure 1. A summary of available functions. Functions marked with a Roman numeral can be invoked at any time. To submit a bid, a user follows in numerical order the functions marked with Arabic numerals. Labels using letters denote internal calls; for example, *1a* is called within function *1*. The label f specifically represents a sub-function called only when some form of malicious user behaviour is observed.

set of auctioneers, a reputation-based mechanism, or a dynamic registration fee for creating auctions. We discuss these design choices further in Section 6.

4.2. Pre-Phase B: Deposit

We now describe how a user $u \in \mathcal{U}$ *registers* for participation in our auction mechanism, in a way that will later allow u to prove eligibility to participate in an auction. The protocol is shown in Algorithm 2.

Merkle-Tree-Based Registry. As shown in function DEPOSIT() of Algorithm 2, user u deposits by committing to a private secret ρ_u, sampled uniformly from $\{0,1\}^\lambda$:

$$C_u = \text{HASH}(\rho_u).$$

The contract $\mathcal{DC}$ records (C_u, D) as in a Merkle-tree-based registry. For a slot s, an observer (user or timestamper) can determine the root of the Merkle tree root_s, as well as all registered users and their commitments, (A_u, C_u). This will later serve as public reference for zero-knowledge membership proofs – users will use root_s to create the proof and timestampers will use the same root_s to verify the proof.

Activity Requirement. Each deposit must remain active by submitting either:

(i) a valid bid commitment, such that the bid is revealed to each timestamper in the reveal phase (winning or losing),
(ii) or a no-operation proof of non-participation,

in every auction until withdrawn. The proofs can be submitted after the auction ends, but being inactive in more than n_A auction triggers can trigger a MAKEINACTIVE() call by an IL Proposer on-chain. Once the user becomes inactive, it must submit a proof of non-participation to regain eligibility. Failure to do so invalidates the deposit, stopping the user from withdrawing the amount. If such proof does not exist (due to misbehaviour), the bidder can never withdraw the deposit, and is effectively slashed.

Withdrawal. A withdrawal can be initiated by referencing the deposit record and signing with the key associated with the address A_u of user u. At that time, the deposit of u must be active, according to the activity requirement conditions, and u must have revealed its bids in all auctions in which it has participated. If u has not participated in some concluded auctions (these can be at most n_A, otherwise MAKEINACTIVE() would have been called, stopping withdrawals), it can submit a proof of non-participation using PROVENOP() (Algorithm 4) for these and for the currently ongoing n_A auctions, verifying that it is not currently using its deposit. The entry for the bidder is instantly removed from the list of bidders, and u is refunded its deposit. We show this in function WITHDRAW().

Inactivity. If u does not revealed its bids (winning or losing) for n_A auctions, then any IL Proposer can call the function MAKEINACTIVE() to start an n_A-long auction window, after which u will be removed from the list of bidders and added to an inactive-bidders list.

During this period, u can call function UPDATEACTIVITY() to directly submit a proof of non-participation for the auctions it did not participate in, or reveal a bid lower than the winning bid for that auction on-chain (we explain in Section 4.4 how the user generates the proof of non-participation Π using function PROVENOP()). Note that during the n_A-auction window, u can still get its bid timestamped, but if u is inactive when the auction ends then the bid can never be revealed as the winning bid.

According to the above, users need to submit proof of non-participation or reveal bids for a certain number of past auctions, at most $3n_A - 1$ (n_A auctions before the window starts, n_A auctions during the window, and $n_A - 1$ other

Algorithm 2 Deposit Registration and Membership Proof

Local for user u:

offline DEPOSIT(D) ▷ Run by u with address A_u
  $\rho_u \leftarrow_r \{0,1\}^\lambda$
  $C_u \leftarrow \text{HASH}(\rho_u)$
  REGISTERDEPOSIT(A_u, C_u, D)

On-chain functions on contract $\mathcal{DC}$:

global deposits ▷ All users' deposits, stored as a merkle tree
global inactives ▷ Users made inactive due to non-participation
global slashed ▷ Slashed deposit handles
global minDeposit ▷ Minimum required deposit
function REGISTERDEPOSIT(A_u, C_u, D)
  require sender $= \mathsf{A}_u$ ▷ caller of this function must be A_u
  require $D \geq$ minDeposit
  Record (A_u, C_u, D)
  Insert (A_u, C_u) in deposits

Called by IL Proposer if the number of not-revealed auctions for user A_u surpasses n_A, i.e., $|\mathbf{U}[u]| \geq n_A$ ($\mathbf{U}$ is defined in Algorithm 4):
function MAKEINACTIVE(A_u)
  require sender $\in \mathcal{P}_{il}$ ▷ only IL Proposers may call this
  require (A_u, C_u) in deposits
  start n_A auction timer
  at end of n_A timer:
    Insert (A_u, C_u) in inactives
    Remove (A_u, C_u) from deposits
function UPDATEACTIVITY(A_u, Π)
  require sender $\in \mathcal{P}_{il} \vee$ sender $= \mathsf{A}_u$
  require inactives contains (A_u, C_u)
  call UPDATEHISTORY(Π)
  Insert (A_u, C_u) in deposits
  Remove (A_u, C_u) from inactives
function SLASHUSR($h_u, (\mathsf{cm}, \pi_u^{\mathsf{m}}), (\mathsf{cm}', \pi_u^{\mathsf{m}'})$)
  require ELGVRF$(h_u, \mathsf{cm}, \pi_u^{\mathsf{m}})$
  require ELGVRF$(h_u, \mathsf{cm}', \pi_u^{\mathsf{m}'})$
  slashed.add(h_u)
function SLASH(A_u, h_u, Π)
  Verify that signature on Π is from A_u
  if $h_u \in \Pi$ and $h_u \in$ slashed **then**
    Remove (A_u, C_u) from deposits
function WITHDRAW(A_u, Π)
  Verify that the message is signed by user
  Verify that signature on Π is from A_u
  UPDATEHISTORY(Π)
  Remove (A_u, C_u) from deposits
  Pay minDeposit to A_u

auctions that might have started before the end of auctions in the n_A auction window). This is re-stated in Lemma B.3.

Slashing users. A user can be slashed if a handle h_u and two different bid commitments $\mathsf{cm}, \mathsf{cm}'$ are given as evidence to function SLASHUSR(), along with verifying proofs $\pi_u^{\mathsf{m}}, \pi_u^{\mathsf{m}\prime}$. This evidence can be provided by timestamperes (see function SIGNCOM() in Algorithm 3). However, the true identity of the user is still unknown at this point, and thus h_u is marked as slashed. Whenever the user tries to use that handle (either to prove NOP or reveal its bid), SLASH is called on the user's address, with requirement that one of the handles that the user sent is in the slashed set.

Security Rationale. This pre-phase will later assist in achieving the following properties: 1) *Duplicate Prevention and Sybil Resistance*: Each bid handle h_u is unique per auction and per deposit, thus the same deposit cannot be used for multiple bids. 2) *Anonymity*: The ZK membership proofs hide A_u and ρ_u while proving that u has a deposit that is active for the duration. 3) *No Free Bid Withdrawal*: Users must either bid or explicitly issue a NOP proof each round, and no user cannot have more than $3n_A - 1$ auctions (from Lemma B.3) for which such a proof does not exist. 4) *Withdrawal Safety*: The enforced waiting period ensures no pending bids remain, hence the user can safely withdraw its deposit.

4.3. Phase 1: Timestamping

Goal. Produce a public, verifiable certificate that a bid commitment existed *before* the auction deadline, without revealing the bid or the bidder's identity.

Inputs and outputs. A user u with address A_u who registered a deposit in Algorithm 2 with secret ρ_u wishes to submit a bid transaction tx for auction AucID.

The output of the phase is a *timestamp certificate*

$$\sigma \;=\; \{(\mathsf{cm}, t_i)_{\sigma_i} : i \in \mathcal{P}_{ts}\},$$

together with its *median timestamp* $\tau = \text{median}(\{t_i : i \in \mathcal{P}_{ts}\})$. In this output, at most f_{ts} timestamps can be from a Byzantine timestamper, thus at least $f_{ts}+1$ timestamps exist from honest timestampers. If a timestamp is not received from a timestamper, then it is considered to be ∞.

Zero-Knowledge Proof of Membership. User u must first prove that it has registered a deposit in the registry with root root_s. We show this in function ELGPRV() in Algorithm 3. Specifically, u creates a pseudonymous *handle* h_u and a commitment cm to its transaction tx (which contains a bid):

$$h_u = \text{HASH}(\rho_u \parallel \mathsf{AucID}),$$
$$\mathsf{cm} = \text{HASH}(tx \parallel C_u).$$

Observe that the handle is unique per auction and deposit, hence the user cannot submit more than one bid for each deposit made in the deposit contract, and that the commitment blinds the actual bid of the user.

Furthermore, u creates a zero-knowledge proof π_u^{m} with public elements (statement) root_s, h_u, and cm, and private

elements (witness) ρ_u, A_u, the Merkle path from root_s to (A_u, C_u), AucID, and tx, attesting that

$$h_u = \textsc{Hash}(\rho_u \parallel \mathsf{AucID}) \ \wedge \ \exists\, C_u : \\ \big(C_u = \textsc{Hash}(\rho_u) \wedge (\mathsf{A}_u, C_u) \in \textsc{Merkle}(\mathsf{root}_s) \\ \wedge \ \mathsf{cm} = \textsc{Hash}(tx \parallel C_u)\big)$$

Essentially this is a proof of membership in root_s. Specifically, the proof demonstrates that: (i) the user knows a secret ρ_u which produces the public handle h_u, (ii) the secret ρ_u corresponds (through the relationship $C_u = \textsc{Hash}(\rho_u)$) to a deposit record (A_u, C_u) in the registry with root root_s, (iii) the commitment cm has been created by hashing some transaction tx and C_u, and (iv) the deposit is active. At the same time, no information about A_u or ρ_u is revealed. Note that AucID is part of the handle, but π_u^{m} does not prove anything about it – we will use this later in Section 4.4.

Protocol for users to obtain timestamp certificate. The user sends the handle h_u, bid commitment cm, and eligibility proof π_u^{m} to the timestampers $\mathcal{P}_{ts}$ through the gossip network, as shown in function SUBMITCOM(). User u waits to collect $2f_{ts}+1$ valid responses from parties in $\mathcal{P}_{ts}$, and for at most $2\Delta_g$ time. After that timeout, u assumes all other parties have sent ∞ as their timestamp. When it receives $2f_{ts}+1$ responses, u runs RECEIVEDTIMESTAMPS() to create the timestamp certificate σ with median timestamp τ, thereby concluding this phase of the protocol.

Protocol for timestampers. Upon receiving such request from a user, the timestampers run the function SIGNCOM() of Algorithm 3, which, if successful, signs the commitment and gossips the timestamp certificate to the user. In order to verify the proof, the timestampers locally run function ELGVRF(), which contains the verification part of the proof constructed in ELGPRV(). It recovers the Merkle root root_s from the slot s and verifies that the proof π_u^{m} is valid for the handle h_u and commitment cm.

Security intuition. Because at most f_{ts} replicas are faulty, the $(f_{ts}+1)$-st order statistic (the median) is within the interval spanned by honest clocks (Lemma B.1). This bounds the median value for all users to be between the timestamps provided by two honest parties and prevents early or late skew beyond a threshold. Since the tuple contains only cm, a handle h_u and is sent back a timestamp τ independent of AucID, the content and identity remain hidden pre-release, while the adversary learns that a bid was placed for one of the n_A (cannot distinguish which) at clock time τ.

4.4. Phase 2: Bid-submission

Goal. Ensure that every *timely* bid (i.e., with $\tau \le t_{\text{end}}$) is considered for the auction outcome and that the *authenticated winner* cannot be censored within the slot. The IL proposers verify these bids, enforce inactivity penalties. Only the winner is ultimately settled on-chain to achieve *Auction Participation Efficiency (APE)*.

Algorithm 3 ObtainTimestamp

Require: User u (with address A_u and deposit secret ρ_u), transaction tx, timestamping validators $\mathcal{P}_{ts}$.
Ensure: Obtain certificate σ and median timestamp τ.

Local for user u:

offline ELGPRV($\rho_u, \mathsf{A}_u, \mathsf{AucID}, tx, s$):
  $C_u \leftarrow \textsc{Hash}(\rho_u)$
  $h_u \leftarrow \textsc{Hash}(\rho_u \parallel \mathsf{AucID})$
  $\mathsf{cm} \leftarrow \textsc{Hash}(tx \parallel C_u)$
  $\mathsf{root}_s \leftarrow$ Merkle root for deposits at slot s
  $\mathsf{path} \leftarrow$ Path from root_s to (A_u, C_u)
  $\pi_u^{\mathsf{m}} \leftarrow \textsc{ZkProve}((\mathsf{root}_s, h_u, \mathsf{cm}), (\rho_u, \mathsf{A}_u, \mathsf{path}, tx))$
  return $(h_u, \mathsf{cm}, \pi_u^{\mathsf{m}})$

offline SUBMITCOM($tx, \rho_u, \mathsf{A}_u, \mathsf{AucID}, s$):
  $(h_u, \mathsf{cm}, \pi_u^{\mathsf{m}}) \leftarrow$ ELGPRV($\rho_u, \mathsf{A}_u, \mathsf{AucID}, tx, s$)
  $(h_u, \mathsf{cm}, \pi_u^{\mathsf{m}}, s) \xrightarrow{\text{GOSSIP}} \mathcal{P}_{ts}$

On receiving $2f_{ts}+1$ responses or waiting for $2\Delta_g$ time from SUBMITCOM*, and padding the timestamps not received with ∞, $\{(\mathsf{cm}, \tau_j)\}_{j\in[2f_{ts}+1]}$:*
offline RECEIVEDTIMESTAMPS()
  $\mathcal{T} \leftarrow \{t_1, \ldots, t_{2f_{ts}+1}\}$
  $\tau \leftarrow \text{median}(\mathcal{T})$
  $\sigma \leftarrow \{(\mathsf{cm}, t_j)_{\sigma_j} : j \in [2f_{ts}+1]\}$.
  return (τ, σ).

Local for timestamper $P_{ts,i}$:

local ids $\leftarrow \{\}$ ▷ Observed handles by $P_{ts,i}$
offline ELGVRF($h_u, \mathsf{cm}, \pi_u^{\mathsf{m}}, s$):
  $\mathsf{root}_s \leftarrow$ Merkle root for deposits at slot s
  require ZKVERIFY($(\mathsf{root}_s, h_u, \mathsf{cm}), \pi_u^{\mathsf{m}}$)

Run by $P_{ts,i}$ on receiving $(h_u, \mathsf{cm}, \pi_u^{m}, s)$:
offline SIGNCOM($\mathsf{cm}, h_u, \pi_u^{\mathsf{m}}, s$)
  $t_i \leftarrow$ local clock time
  require ELGVRF($h_u, \mathsf{cm}, \pi_u^{\mathsf{m}}, s$)
  if ids$[h_u] = (\mathsf{cm}' \neq \mathsf{cm}, \pi_u^{\mathsf{m}}, s')$ **then**
    SLASHUSR($h_u, (\mathsf{cm}, \pi_u^{\mathsf{m}}, s), (\mathsf{cm}', \pi_u^{\mathsf{m}'}, s')$)
  else
    ids.add($h_u, \mathsf{cm}, \pi_u^{\mathsf{m}}, s$)
    $(\mathsf{cm}, t_i)_{\sigma_i} \xrightarrow{\text{GOSSIP}} u$ ▷ σ_i denotes signature by $P_{ts,i}$.

Inputs and Outputs. The reveal phase takes as input the user's bid transaction tx for a given auction AucID and the timestamp certificate (τ, σ) obtained during Phase 1. To make the proof for inactivity, the user inputs AucID_ℓ, the last auction the user revealed for. On the IL proposer's side, the inputs are the message $(tx, \mathsf{AucID}, (\tau, \sigma), \Pi)$ received from the user, the local record of valid bids $\mathcal{B}$, the user activity mapping $\mathbf{U}$, and the global registries of active and slashed deposits $(\mathsf{deposits}, \mathsf{slashed})$.

At the end of this phase, all honest IL Proposers would have a set of valid bids $\mathcal{B}$, on top of which any auction logic can be run. The local winning bids are included in the Inclusion List for the reserved slot S for the auction.

Overview. Before presenting the pseudocode in Algorithm 4, we provide an overview. Each user who participated in auction AucID submits its bid transaction tx together with its timestamp certificate (τ, σ) and a proof of inactivity Π. The proof of inactivity serves two purposes:

1) it demonstrates that, for all auctions between the user's last revealed auction AucID_ℓ and the current one, the user either revealed a valid bid or explicitly declared a no-operation (NOP); and
2) it enables IL proposers to detect users who attempt to hide activity or submit multiple bids, allowing timestampers to *slash* the associated deposits.

Once received, IL proposers verify the proof, update user-activity records, and include the bid in their local set of bids $\mathcal{B}$. The winning bid – according to some predefined auction logic – is included in their Inclusion Lists. Only those bids, whose median timestamp τ precedes the auction's end time t_{end}, and that are revealed to the IL Proposer before slot S, are considered. The inclusion of the winning bid in all honest ILs guarantees censorship resistance within the slot.

To prove that the bid presented was generated for the particular auction ID, we require the u to create another zero-knowledge proof π_u^{a} with public elements (statement) h_u, AucID, C_u and private elements (witness) ρ_u, attesting

$$h_u = \textsc{Hash}(\rho_u \,\|\mathsf{AucID}) \;\wedge\; \big(C_u = \textsc{Hash}(\rho_u)\big)$$

Along with this, the user must also send a history of handles it can generate for all auctions it did not participate in. This is done through a NOP transaction, which the IL Proposer gossips to the timestampers, so they can check for any repetition of handle. The IL Proposer then updates its local view of user's activity .

FOCIL in our construction. We use a construction similar to FOCIL (Section 3.1) to obtain censorship resistance against a malicious auctioneer, with the following modifications. First, the protocol is equipped with a validity predicate, and a transaction can only be included in an IL if it satisfies the predicate. A transaction-certificate tuple (tx, τ, σ) is valid if it satisfies the following conditions:

- All signatures in (σ) are valid and correspond to the same commitment (cm) and timestamp values are valid.
- The median timestamp τ is within the valid window for the current slot; specifically, the network enforces that $\tau \leq t$, where t denotes the cutoff time to submit bids.
- The bid satisfies auction rules (e.g., valid fee, bid format).

If these conditions hold, an IL Proposer adds tx to its IL, otherwise tx is ignored. The timestamp certificate σ ensures that only transactions created before the cutoff t are accepted.

Second, an IL in our construction does not contain all transactions received by its proposer but only the one – the winning bid. This also implies that we define no limit size for an IL, as it only contains one transaction. Finally, the rest of the parties relevant in FOCIL, i.e., validators, block builder, and attesters, are assumed to be part of underlying blockchain protocol. We assume the following fork-choice rule: If a set of rules defined by auctions logic is not

Algorithm 4 BidReveal

Require: auction AucID with end time t_{end}, transaction tx, IL Proposers $\mathcal{P}_{il}$
Ensure: IL contains $b_w^{\mathsf{AucID}} = \max(\mathcal{B})$ for slot S

Local for user u:

local AucID_ℓ ▷ Auction last revealed for

At time $t \geq t_{\text{end}} + \Delta_g$ run:

offline REVEALBID$(tx, \mathsf{AucID}, (\tau, \sigma))$
  $\Pi \leftarrow$ PROVENOP(AucID)
  $\pi_u^{\mathsf{a}} \leftarrow$ ZKPROVE$((h_u, \mathsf{AucID}, C_u), \rho_u)$
  $\forall\, P_{il} \in \mathcal{P}_{il} : (tx, \mathsf{AucID}, (\tau, \sigma), \pi_u^{\mathsf{a}}, \Pi) \xrightarrow{\text{DIRECT}} P_{il}$
  $\mathsf{AucID}_\ell \leftarrow \mathsf{AucID}$

offline PROVENOP(AucID)
  $\mathsf{AucID}_{max} \leftarrow \min(\mathsf{AucID}, \mathsf{AucID}_\ell + 3n_A - 1)$
  $\Pi \leftarrow []$
  for $i \in [\mathsf{AucID}_\ell, \mathsf{AucID}_{max}]$ **do**
    $(h_u^i, \mathsf{cm}^i, \pi_u^{\mathsf{m},i}) \leftarrow$ ELGPRV$(\rho_u, \mathsf{A}_u, i, \mathsf{NOP},\ s)$
    Π.append$(i, h_u^i, \mathsf{cm}^i, \pi_u^{\mathsf{m},i}, \mathsf{NOP})$
  return $(\Pi)_{\sigma_u}$ ▷ signed proof

Local for IL Proposer P_{il}:

local $\mathcal{B} \leftarrow \{\}$ ▷ valid bids
local $\mathbf{U}$ ▷ Mapping from $\mathcal{U}$ to AucIDs for which user has not revealed.

On receiving $(tx, \mathsf{AucID}, (\tau, \sigma), \pi_u^{\mathsf{a}}, \Pi)$ from user u:

offline PROCESSBID$(tx, \mathsf{AucID}, (\tau, \sigma), \pi_u^{\mathsf{a}}, \Pi)$
  require ZKVERIFY$((h_u, \mathsf{AucID}, C_u), \pi_u^{\mathsf{a}})$
  UPDATEHISTORY(Π)
  for $(i, h_u^i, \mathsf{cm}^i, \pi_u^{\mathsf{m},i}, \mathsf{NOP}) \in \Pi$ **do**
    if $h_u^i \in$ slashed **then**
      call SLASH (A_u, C_u) on-chain
      abort
  require $\mathsf{A}_u \in$ deposits ▷ deposits at current s
  $(\_, \_, \_, t_{\text{end}}, S) \leftarrow$ GETAUCTIONRECORD(AucID)
  if $\tau \leq t_{\text{end}} \wedge$ clock time $\leq S$.time **then**
    require verify certificate σ
    $\mathcal{B}$.insert(tx)
  $\mathbf{U}[u]$.remove(AucID)

offline UPDATEHISTORY(A_u, Π)
  Verify signature on Π
  for $(i, h_u^i, \mathsf{cm}^i, \pi_u^{\mathsf{m},i}, \text{NOP}) \in \Pi$ **do**
    $(h_u^i, (\mathsf{NOP}, \mathsf{cm}^i, \pi_u^{\mathsf{m},i})) \xrightarrow{\text{GOSSIP}} \mathcal{P}_{ts}$
  wait for $4\Delta_g$ ▷ Allow $P_{ts,}$ to slash
  for $(i, h_u^i, \mathsf{cm}^i, \pi_u^{\mathsf{m},i}, \mathsf{NOP}) \in \Pi$ **do**
    $\mathbf{U}[u]$.remove(i)

Description: In slot S of AucID, each IL Proposer runs the auction logic on $\mathcal{B}$ and includes the winner in the local inclusion list. Users that did not reveal get AucID added to set $\mathbf{U}$. The proposer for a PoS blockchain only needs to includ the winning bid among all local inclusion lists. This is then verified by the attesters in the PoS chain.

followed by the proposer on all declared inclusion lists (which are singleton for each auction ID) of setbids before, then fork choice invalidates the block (i.e., the consensus rejects the block, and the next block will be created as if previous block did not arrive). Note that we assume that if an IL Proposer declares tx_1 as its local winning bid, and another IL Proposer declares tx_2, then the proposer (and attesters) must be able to identify which transaction is the correct winning bid.

Protocol for users. We now present the construction in Algorithm 4. The entry point for a user is function REVEALBID(), which sends $(tx, \mathsf{AucID}, (\tau, \sigma), \pi_u^{\mathsf{a}}, \Pi)$ to each IL Proposer. Here, π_u^{a} is a proof that the AucID is the same as the one used in h_u. Π contains proofs for *previous* auctions, which the user must submit in case it did not participate in some of them. These proofs can be generated by function PROVENOP().

The PROVENOP() function locally remembers the last auction AucID_ℓ for which the user revealed a bid. It then generates proofs for all auction numbers i between AucID_ℓ and the current auction AucID (and for at most $3n_A-1$ after AucID_ℓ, which represents the maximum number of auctions the user could have received a timestamp for without being considered inactive.). Each of those proofs is exactly the membership proof for auction i, as presented in Section 4.3, but for a specific transaction, denoted as NOP, indicating that user did not participate in auction i.

Protocol for IL Proposers. An IL Proposer P_{il} handles the received $(tx, \mathsf{AucID}, (\tau, \sigma), \pi_u^{\mathsf{a}}, \Pi)$ from a user as shown in function PROCESSBID(). The first step for P_{il} is to forward all the proofs of inactivity Π to the timestampers (who process them using the function ELGVRF() of Algorithm 2 as described in the previous section). Then the IL Proposer waits for the timestampers to process the proofs, which may result in the user being slashed (if the timestampers detect that the user has submitted a different bid commitment for the same handle). If this is the case, the IL Proposer does not further process the received transaction.

The next step for P_{il} is to verify the proof π_u^{a} and the certificate σ (as described previously). If succesfull, P_{il} inserts the transaction tx into its set of valid bids $\mathcal{B}$.

Keeping users' deposit active. Finally, we explain how users can make sure their deposit remains active. If a user is actively sending revealed bids to the IL Proposer, then each honest IL Proposer would have the history of handles generated by the user. This implies that the user would be active for the honest IL Proposer. A malicious IL Proposer can trigger the inactivity notice for any user; however, given the n_A window in which the user's deposit is still active, the user can send a reveal to any honest IL Proposer that can update user's activity on-chain. Alternatively, the user can prove its activity or Non Participation directly on-chain.

Re-activating users' deposit. In case the user goes offline or sparringly participates in auctions, any IL Proposer could make the user's deposit inactive. Being inactive helps the user to not prove its entire history when it comes back online; instead it just has to prove its non-participation in $3n_A-1$ auctions since it went offline. This number is derived from the fact that any user can have at most $3n_A-1$ auctions that it has not revealed in before being added to inactives set (Lemma B.3).

4.5. Security Analysis

Definition 8 (Security assumptions)**.** *In all the following lemmas and theorems, we assume the following security assumptions:*

- *The network is synchronous with a delay of at most* Δ.
- *All parties have a* $\Delta-$ *synchronized clock.*
- *The gossip network has a delay of at most* Δ_g
- *Static corruption with at most* f_{ts} *Byzantine replicas in* $\mathcal{P}_{ts}$*, the rest being honest.*
- *Static corruption with existential honesty in* $\mathcal{P}_{il}$
- *Implicitly assume that the majority of attesters in the Proof-of-stake chain are honest. This assumption is required to use FOCIL as a subroutine.*

Under the above assumptions, we prove the following theorem statements in Section B.

Theorem 4.1 (Censorship Resistance)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time* t_{end}*, and has some pre-defined settlement slot* S*. The auction satisfies* $\delta-$SST Censorship Resistance with $\delta = (\Delta_g + \Delta)$ *(according to Definition 2).*

Theorem 4.2 ((Relaxed) Hiding)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time* t_{end}*. Let* AUCSET(τ) *represent the ongoing parallel auctions at clock time* τ*. Let* $\mathcal{U} = \{\mathsf{A}_u : (\mathsf{A}_u, \_) \in \mathsf{deposits}\}$ *represent the users in* deposits *in the slot right before clock time* τ*. The auction satisfies the following items from the* Relaxed t_h-Strong Hiding *property (Definition 3): (i) Indistinguishability (ii) Existential Obfusction within* AUCSET(τ) *(iii) User Obfuscation within* $\mathcal{U}$*.*

Theorem 4.3 (Auction Participation Efficiency)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time* t_{end}*, and has some pre-defined settlement slot* S*. The cost to an honest active user if it does not win an auction tends to* 0*.*

Theorem 4.4 (No Free Bid Withdrawal)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time* t_{end}*, and has some pre-defined settlement slot* S*. If* $\mathsf{minDeposit} \geq (3n_A - 1) \cdot u$*, meaning that, by not revealing a commited bid, the adversary pays an amount of at least* u *per auction, the auction satisfies the* No Free Bid Withdrawal *property.*

5. Related Work

Auctions are a foundational economic tool, and blockchains offer new environments in which to implement them. Classical auction formats like English, Dutch, first-price, and second-price auctions have been adapted

for decentralized settings, but these adaptations come with challenges. This section surveys auction mechanisms implemented on blockchain, focusing on designs relevant to sealed-bid auctions, off-chain execution, and auctions for protocol-level resources like blockspace.

5.1. Open On-Chain Auctions

Open auctions, most commonly in the English and Dutch formats, are widely used in blockchain systems because they are simple to implement and easy to verify on-chain. In an English auction, described formally by Milgrom and Weber [9], bids increase incrementally until the auction closes, and the highest bidder wins. Despite their simplicity, these mechanisms are prone to issues such as sniping and frontrunning, since all pending bids are visible on the network. To address these problems, some platforms employ randomized or extended closing times, as in the candle auction model adopted by Polkadot [10].

Dutch auctions, introduced in classical auction theory as descending-price mechanisms [11], require less interaction and typically settle more quickly. MakerDAO, for instance, transitioned from an English to a Dutch auction in its liquidation system to improve reliability during periods of network congestion [12]. However, Dutch auctions still reveal price movements and timing on-chain, leaving them exposed to strategic bidding and potential manipulation by miners or validators.

Although open on-chain auctions offer transparency and composability, they inevitably expose bid amounts and timing on the public ledger. As a result, they provide limited SST censorship resistance, and achieving fairness in adversarial environments remains challenging even with measures such as randomized closing rules, auction extensions, or alternative transaction ordering schemes.

5.2. Commit-and-Reveal Sealed-Bid Auctions

Sealed-bid auctions are more difficult to implement on-chain because of blockchain transparency. First-price and second-price formats, including the Vickrey auction [11], rely on bids remaining secret until they are revealed. The standard approach to achieve this is the commit-and-reveal model, where participants first submit a hash of their bid and then reveal the bid and nonce in a later phase. This design preserves confidentiality during the commitment phase while allowing public verification during the reveal.

The commit-and-reveal mechanism, however, introduces latency through multiple transaction rounds and additional participation costs from inclusion fees. Early implementations, such as the Ethereum Name Service (ENS) auction system [13], demonstrate both the practicality and the limitations of this approach, as the original auction design has since been deprecated [14]. ENS and related systems [15]–[18] extend this model to improve reliability and discourage aborts, while others use stronger cryptographic verification to achieve similar guarantees [18].

The commit-and-reveal model remains costly and operationally inefficient. Losing bidders still pay inclusion fees, and strategic participants can delay or withhold reveals to influence outcomes. When deadlines are known in advance, bids often cluster near the closing time as participants incorporate external information. This gives the block proposer temporary control over inclusion, enabling selective censorship or delayed acceptance, particularly in time-sensitive financial settings. Even with commitment schemes, on-chain timing can still leak or bias information about bidding behavior. Our work avoids these issues by removing the need for on-chain reveals and preventing block proposers from influencing bid inclusion.

5.3. Cryptographic and TEE-Based Designs

Several designs extend the basic commit-and-reveal model using cryptographic techniques to improve privacy and fairness. Encrypted mempools, which conceal transaction contents until inclusion, have been proposed to enhance bid confidentiality [19]–[24]. FAST [25] implements first- and second-price sealed-bid auctions using secret cryptocurrency transactions and public smart contracts, preserving losing-bid privacy and enforcing fairness through collateralized penalties. It assumes confidential-transaction support rather than protocol changes, making it complementary to encrypted-mempool approaches.

Several cryptographic designs [26]–[30] employ semi-trusted auctioneers that collect encrypted bids and compute outcomes under verifiable proofs, ensuring correctness while limiting information leakage. TEE-based systems [31] remove the auctioneer but shift trust to hardware enclaves that execute auctions privately with attestable results. Multi-party computation frameworks such as STRAIN [32] distribute computation across participants, avoiding any single point of control but increasing coordination cost. Riggs [4] uses time-lock puzzles and solver incentives to ensure bids are eventually revealed without user interaction, maintaining liveness and decentralization at the cost of computation.

Cryptobazaar [6] achieves similar privacy guarantees entirely at the application layer by submitting encrypted bids and proving correctness on-chain, revealing only the winning bid at settlement. In contrast to our work, the protocol has weaker hiding gurantees, lower efficiency, does not address SST censorship resistance. ZeroAuction [5] adopts an execution-layer approach, using delayed decryption of encrypted bids to merge commitment and revelation into a single phase. This removes deposits and extra interactions but requires protocol support for delayed execution and threshold decryption. Some on-chain limits remain, including SST censorship risk and inclusion costs for losing bids. Our mechanism mitigates these by curbing proposer influence and recording only the winning bid on-chain.

5.4. Off-Chain Auctions

To reduce costs and improve user experience, many auction systems move bidding logic off-chain. NFT mar-

ketplaces like OpenSea use off-chain signed bids [33], with only the final accepted bid settled on-chain. This cuts gas costs but reintroduces centralized trust, since order relays decide which bids get matched. CoW Protocol takes a similar approach: users submit signed orders to an off-chain orderbook, solvers compete to find a clearing solution, and only the winning settlement is posted on-chain [34].

Similar market structures exist at the blockchain infrastructure layer. Early Ethereum users engaged in *priority gas auctions*, where transactions competed for inclusion by bidding up gas prices to miners [35]. To reduce waste and frontrunning, *Flashbots* introduced a private off-chain auction where searchers submitted transaction bundles directly to miners, with bids paid as explicit transfers rather than gas price competition [36]. This mechanism later evolved into *Proposer-Builder Separation (PBS)*, in which specialized builders construct blocks and submit sealed bids to proposers for the right to publish them. PBS, implemented through MEV-Boost relays [37], formalizes blockspace as a recurring first-price auction executed every slot.

While off-chain auctions scale well, they rely on intermediaries that can selectively censor or delay bids. Our work removes this dependency by operating without any single trusted party.

5.5. Protocol Layer Censorship Resistance

Recent work has explored protocol-level mechanisms to strengthen censorship resistance. FOCIL (Fork-Choice-Enforced Inclusion Lists) [2], [8] introduces a committee-based inclusion scheme where a subset of validators publishes local inclusion lists that proposers must aggregate and include. It provides conditional protection assuming existential honesty within the inclusion-list committee, but remains vulnerable to block-filling, as proposers can crowd out listed transactions by saturating available block space. AUCIL (Auction-Based Inclusion List) [3] extends this idea in a rational model, using an auction over inclusion lists to achieve unconditional inclusion guarantees but still leaves the proposer with a latency advantage.

Multiple Concurrent Proposers (MCP) [1] proposes a more invasive architectural change in which several proposers assemble blocks in parallel, providing stronger SST censorship but at the cost of higher protocol complexity and worse trust assumptions. While MCP offers the strongest guarantees, its invasive design may limit applicability in systems that require SST censorship resistance without modifying the consensus protocol. Our work addresses this gap by providing auction-level SST censorship resistance within the existing protocol architecture.

5.6. Rational-Auction Protocols.

A related line of work studies sealed-bid auctions in the *rational* adversarial model. Ganesh et al. [38] present a protocol for *Vickrey* auctions that achieves incentive compatibility among rational parties using cryptographic commitments and game-theoretic punishments. Earlier, Ganesh et al. [39] propose concretely efficient protocols for *first-price* auctions secure in the presence of rational bidders, focusing on equilibrium robustness and verifiable correctness under deviation costs. In contrast, our construction targets *permissionless* deployment and addresses two blockchain-specific challenges absent from prior work: *SST censorship* at inclusion time and *timing leakage* about bid existence. We mitigate these through (i) timestamp certificates ensuring *existential indistinguishability* of bids and (ii) inclusion lists for SST censorship resistance, avoiding coordinated decryption committees or multi-round reveals. Our *APE* objective further minimizes losing-bidder fees, whereas rational-auction protocols focus on strategic correctness under different participation assumptions.

6. Conclusion and Future Work

Our protocol fits into a broader effort to make sealed-bid auctions practical on blockchains. Conventional sealed-bid designs do not address SST censorship: because the block proposer has monopoly control for the duration of its slot, a malicious proposer can suppress timely honest bids or exploit its latency advantage to react strategically to information that other bidders where not able to not mitigate the underlying latency advantage. More recent approaches remove the proposer monopoly by employing multiple concurrent proposers, but doing so demands substantial modifications to consensus, introduces considerable protocol complexity, and impose a significant on-chain load.

Our work takes a different direction. We design a lightweight, consensus-adjacent mechanism that achieves the strengthened properties required for high-value, time-sensitive auctions. In particular, our protocol provides (i) δ-selective short-term censorship resistance, (ii) relaxed t_h-strong hiding, (iii) no free bid withdrawal, and (iv) auction participation efficiency. Importantly, because most of the mechanism operates off-chain, it places minimal demand on blockspace, though this necessitates a more involved deposit mechanism and eligibility proofs.

Future Work. First, a key direction for future work is designing incentive mechanisms for timestampers. Our protocol currently relies on an honest majority among timestampers to produce correct and timely certificates; replacing this assumption with explicit economic incentives would provide stronger robustness and make the timestamping layer secure even under strategic behavior. A related question exists for IL proposers, though it is less pressing in our setting – since we require only existential honesty – and prior work has already made progress on incentivizing IL proposers [3].

Second, while our protocol currently reserves a fixed amount of capacity in every slot, a more efficient approach would allow auctioneers to explicitly reserve capacity only for the slots in which their auctions run. Auctioneers could also reserve several consecutive slots, ensuring guaranteed space even if a proposer fails to produce a block in one of them. Such a mechanism would provide proposers with direct compensation for guaranteeing space. Designing a

reservation system that is secure and incentive compatible remains an open challenge.

Third, our protocol does not yet include a Sybil-resistance mechanism for auctioneers to prevent malicious parties from spamming many low-value auctions. This is closely related to the previous point: the reservation fees used for slot reservations could themselves serve as a Sybil-resistance mechanism, limiting auction creation to auctioneers willing to pay for capacity. More generally, lightweight fee- or deposit-based eligibility rules, or the simpler option of treating auctioneers as a permissioned set, remain important directions for future work.

Finally, a full economic analysis remains open. This includes determining appropriate penalties for non-reveals; and, if a slot-reservation mechanism is introduced, setting reservation fees appropiately to incentivize block proposers to include auction-settlement transactions. The same analysis applies to any auction-registration mechanism, where fees must deter spam while remaining compatible with auctioneer incentives. Moreover, if timestampers are made rational rather than honest, the corresponding incentive structure for correct and timely timestamping must also be analysed.

References

[1] P. Garimidi, J. Neu, and M. Resnick, *Multiple concurrent proposers: Why and how*, 2025. arXiv: 2509.23984 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2509.23984.

[2] T. Thiery, B. Monnot, F. D'Amato, and J. Ma, *Fork-Choice-Enforced Inclusion Lists (FOCIL): A Simple Committee-Based Inclusion List Proposal*, https://ethresear.ch/t/fork-choice-enforced-inclusion-lists-focil-a-simple-committee-based-inclusion-list-proposal/19870, Ethereum Research, 2024.

[3] S. Wadhwa, J. Ma, T. Thiery, *et al.*, *AUCIL: An Inclusion List Design for Rational Parties*, Cryptology ePrint Archive, Paper 2025/194, 2025. [Online]. Available: https://eprint.iacr.org/2025/194.

[4] N. Tyagi, A. Arun, C. Freitag, R. Wahby, J. Bonneau, and D. Mazières, *Riggs: Decentralized sealed-bid auctions*, Cryptology ePrint Archive, Paper 2023/1336, 2023. [Online]. Available: https://eprint.iacr.org/2023/1336.

[5] H. Zhang, M. Yeo, V. Estrada-Galiñanes, and B. Ford, "Zeroauction: Zero-deposit sealed-bid auction via delayed execution," in *Financial Cryptography and Data Security. FC 2024 International Workshops*, Springer, 2024, pp. 170–188. DOI: 10.1007/978-3-031-69231-4_12. [Online]. Available: https://doi.org/10.1007/978-3-031-69231-4_12.

[6] A. Novakovic, A. Kavousi, K. Gurkan, and P. Jovanovic, *Cryptobazaar: Private sealed-bid auctions at scale*, Cryptology ePrint Archive, Paper 2024/1410, 2024. [Online]. Available: https://eprint.iacr.org/2024/1410.pdf.

[7] E. Fox, M. Pai, and M. Resnick, *Censorship resistance in on-chain auctions*, 2023. arXiv: 2301.13321 [econ.TH]. [Online]. Available: https://arxiv.org/abs/2301.13321.

[8] E. I. Proposals, *Eip-7805: Fork-choice enforced inclusion lists (focil) [draft]*, en-US, Nov. 2024. [Online]. Available: https://eips.ethereum.org/EIPS/eip-7805.

[9] P. R. Milgrom and R. J. Weber, "A theory of auctions and competitive bidding," *Econometrica*, vol. 50, no. 5, pp. 1089–1122, 1982. [Online]. Available: https://cramton.umd.edu/market-design-papers/milgrom-weber-a-theory-of-auctions-and-competitive-bidding.pdf.

[10] *Polkadot parachain slot auctions explained*, https://support.polkadot.network/support/solutions/articles/65000182287-how-do-parachain-slot-auctions-work-, Accessed: 2025-11-01.

[11] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961. [Online]. Available: https://www.jstor.org/stable/2977633.

[12] *MakerDAO Liquidations 2.0*, https://docs.makerdao.com/smart-contract-modules/dog-and-clipper-detailed-documentation, Accessed: 2025-11-01.

[13] *ENS: A beginner's guide to buying an ENS domain*, https://weka.medium.com/a-beginners-guide-to-buying-an-ens-domain-3ccac2bdc770, Accessed: 2025-11-01.

[14] *ENS: Refund of unrevealed bids*, https://support.ens.domains/en/articles/9739338-reclaim-app-reclaim-ens-domains-claim-deposits-from-2017-ens-auctions, Accessed: 2025-11-01.

[15] C. Braghin, S. Cimato, E. Damiani, and M. Baronchelli, "Designing smart-contract based auctions," in *Security with Intelligent Computing and Big-data Services*, Springer, 2018, pp. 180–191. DOI: 10.1007/978-3-030-16946-6_5. [Online]. Available: https://doi.org/10.1007/978-3-030-16946-6_5.

[16] M. Król, A. Sonnino, A. G. Tasiopoulos, I. Psaras, and E. Rivière, "PASTRAMI: privacy-preserving, auditable, scalable & trustworthy auctions for multiple items," vol. abs/2004.06403, 2020. arXiv: 2004.06403. [Online]. Available: https://arxiv.org/abs/2004.06403.

[17] G. Lu, Y. Zhang, Z. Lu, J. Shao, and G. Wei, "Blockchain-based sealed-bid domain name auction protocol," in *Applied Cryptography in Computer and Communications*, B. Chen and X. Huang, Eds., Cham: Springer International Publishing, 2021, pp. 25–38, ISBN: 978-3-030-80851-8.

[18] B. Chen, X. Li, T. Xiang, and P. Wang, "SBRAC: Blockchain-based sealed-bid auction with bidding price privacy and public verifiability," *Journal of Information Security and Applications*, vol. 68, p. 103 235, 2022. DOI: 10.1016/j.jisa.2021.103082. [Online]. Available: https://doi.org/10.1016/j.jisa.2021.103082.

[19] R. Fernando, G.-V. Policharla, A. Tonkikh, and Z. Xiang, *TrX: Encrypted Mempools in High Performance BFT Protocols*, Cryptology ePrint Archive, Paper 2025/077, 2025. [Online]. Available: https://eprint.iacr.org/2025/2032.

[20] A. R. Choudhuri, S. Garg, G. V. Policharla, and M. Wang, "Practical mempool privacy via one-time setup batched threshold encryption," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity25/sec25cycle1-prepub-1216-choudhuri.pdf.

[21] D. Boneh, E. Laufer, and E. N. Tas, *Batch decryption without epochs and its application to encrypted mempools*, Cryptology ePrint Archive, Paper 2025/1254, 2024. [Online]. Available: https://eprint.iacr.org/2025/1254.

[22] A. R. Choudhuri, S. Garg, J. Piet, and G.-V. Policharla, "Mempool privacy via batched threshold encryption: Attacks and defenses," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 3513–3529. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri.

[23] A. Agarwal, R. Fernando, and B. Pinkas, "Efficiently-thresholdizable batched identity based encryption, with applications," in *Annual International Cryptology Conference*, 2025.

[24] J. Bormet, S. Faust, H. Othman, and Z. Qu, "BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025. [Online]. Available: https://www.usenix.org/conference/usenixsecurity25/presentation/bormet.

[25] B. David, L. Gentile, and M. Pourpouneh, "FAST: Fair Auctions via Secret Transactions," in *Applied Cryptography and Network Security*, Springer, 2022, pp. 727–747. DOI: 10.1007/978-3-031-09234-3_36. [Online]. Available: https://doi.org/10.1007/978-3-031-09234-3_36.

[26] H. Abulkasim, A. Mashatan, and S. Ghose, "Quantum-based privacy-preserving sealed-bid auction on the blockchain," *Optik*, vol. 242, p. 167 039, 2021, ISSN: 0030-4026. DOI: https://doi.org/10.1016/j.ijleo.2021.167039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402621007233.

[27] T. Constantinides and J. Cartlidge, "Block auction: A general blockchain protocol for privacy-preserving and verifiable periodic double auctions," in *2021 IEEE International Conference on Blockchain (Blockchain)*, 2021, pp. 513–520. DOI: 10.1109/Blockchain53845.2021.00078.

[28] H. Desai, M. Kantarcioglu, and L. Kagal, "A hybrid blockchain architecture for privacy-enabled and accountable auctions," in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 34–43. DOI: 10.1109/Blockchain.2019.00014. [Online]. Available: https://doi.org/10.1109/Blockchain.2019.00014.

[29] H. S. Galal and A. M. Youssef, "Verifiable sealed-bid auction on the ethereum blockchain," in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, Nieuwpoort, Curaçao: Springer-Verlag, 2018, pp. 265–278, ISBN: 978-3-662-58819-2. DOI: 10.1007/978-3-662-58820-8_18. [Online]. Available: https://doi.org/10.1007/978-3-662-58820-8_18.

[30] G. Sharma, D. Verstraeten, V. Saraswat, J.-M. Dricot, and O. Markowitch, "Anonymous sealed-bid auction on ethereum," *Electronics*, vol. 10, no. 19, p. 2340, 2021. DOI: 10.3390/electronics10192340. [Online]. Available: https://doi.org/10.3390/electronics10192340.

[31] H. Desai and M. Kantarcioglu, "Secauctee: Securing auction smart contracts using trusted execution environments," in *2021 IEEE International Conference on Blockchain (Blockchain)*, 2021, pp. 448–455. DOI: 10.1109/Blockchain53845.2021.00069.

[32] E.-O. Blass and F. Kerschbaum, "Strain: A secure auction for blockchains," in *Computer Security – ESORICS 2018*, Springer, 2018, pp. 87–110. DOI: 10.1007/978-3-319-99073-6_5. [Online]. Available: https://eprint.iacr.org/2017/1044.

[33] Lunaray, *How nfts are traded on opensea and how hackers can transfer your nfts*, https://medium.com/coinmonks/how-nfts-are-traded-on-opensea-and-how-hackers-can-transfer-your-nfts-c491455087, Accessed: 2025-11-07, 2022.

[34] *CoW Protocol Solver Auctions*, https://docs.cow.fi/, Accessed: 2025-11-01.

[35] P. Daian, S. Goldfeder, T. Kell, *et al.*, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE symposium on security and privacy (SP)*, IEEE, 2020, pp. 910–927.

[36] *Flashbots: Frontrunning the MEV Crisis*, https://writings.flashbots.net/frontrunning-mev-crisis, Accessed: 2025-11-07, 2021.

[37] *MEV-Boost Architecture Overview*, https://boost.flashbots.net/, Accessed: 2025-11-01.

[38] C. Ganesh, S. Gupta, B. Kanukurthi, and G. Shankar, "Secure vickrey auctions with rational parties," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4062–4076. DOI: 10.1145/3658644.3670311. [Online]. Available: https://doi.org/10.1145/3658644.3670311.

[39] C. Ganesh, B. Kanukurthi, and G. Shankar, "Secure auctions in the presence of rational adversaries," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1173–1186. DOI: 10.1145/3548606.3560706. [Online]. Available: https://doi.org/10.1145/3548606.3560706.

Appendix A.
LLM Usage

LLMs were used for editorial purposes in this manuscript, and all outputs were inspected by the authors to ensure accuracy and originality.

Appendix B.
Security Analysis

B.1. Intermediate results

We first prove some intermediate results, which we then use to prove the main theorems for our construction.

Lemma B.1 (Median robustness of timestamp certificate)**.** *Let $\mathcal{P}_{ts}$ be a committee of timestampers with at most f_{ts} corrupted, where $|\mathcal{P}_{ts}| = 2f_{ts}+1$. Let $\mathcal{T} = \{\tau_i : i \in \mathcal{P}_{ts}\}$ be the set of timestamps that the user collects on the commitment* cm*, where* at most *f_{ts} timestamps of $\mathcal{T}$ may be adversarially chosen. Given Δ_g as the maximum delay of the gossip network and $\Delta-$synchronized clocks, and assuming that the user treats any missing stamps at timeout as ∞. Let τ be the median of $\mathcal{T}$, and $\mathcal{T}_{\mathsf{hon}}$ represents the subset received from honest parties. Then:*

1) ***Honest span.***

$$\min(\mathcal{T}_{\mathsf{hon}}) \le \tau \le \max(\mathcal{T}_{\mathsf{hon}}),$$

In particular, the adversary cannot move the median outside the honest span.

2) ***User drift bound.*** *If t_u represents the time at which user sends its transaction*

$$\max(\mathcal{T}_{\mathsf{hon}}) - t_u \le \Delta_g + \Delta,$$
$$\min(\mathcal{T}_{\mathsf{hon}}) - t_u \ge \Delta_g - \Delta,$$

3) ***Drift bound.***

$$\max(\mathcal{T}_{\mathsf{hon}}) - \min(\mathcal{T}_{\mathsf{hon}}) \le 2\Delta,$$

hence τ deviates from any honest stamp by at most $\Delta_g + \Delta$.

Proof. Honest span. We prove this statement by Pigeon Hole Principle. There exist f_{ts} values before and after the median timestamp. Since we have $f_{ts}+1$ honest timestamps, either the median timestamp is honest, or at least one of the timestamps before or after the median is filled with an honest timestamp. Thus $\min(\mathcal{T}_{\mathsf{hon}}) \le \tau$ and $\max(\mathcal{T}_{\mathsf{hon}}) \ge \tau$. □

User drift bound. If the user sends the transaction at their local clocktime t_u, then from the perspective of an honest timestamper ($P_{ts,i}$), the sending time of the transaction (accounting for the clock drift) must be

$$t_u - \Delta \le t_u^{P_{ts,i}} \le t_u + \Delta.$$

The time it receives the transaction and assigns it a timestamp is thus

$$t_u - \Delta + \Delta_g \le \tau_i = (t_u^{P_{ts,i}} + \Delta_g) \le t_u + \Delta + \Delta_g.$$

Thus, for all honest $P_{ts,i}$, the assigned timestamp is given by the previous equation. In other words,

$$\max(\mathcal{T}_{\mathsf{hon}}) \le t_u + \Delta + \Delta_g \tag{1}$$

and

$$\min(\mathcal{T}_{\mathsf{hon}}) \ge t_u - \Delta + \Delta_g \tag{2}$$

which proves the statement. □

Drift bound. Subtracting Equations (1) and (2), we get

$$\max(\mathcal{T}_{\mathsf{hon}}) - \min(\mathcal{T}_{\mathsf{hon}}) \le 2\Delta$$

□

Lemma B.2 (Completeness of membership proofs)**.** *An honest user can always create a statement* $(\mathsf{root}_s, h_u, \mathsf{cm})$ *and verifying membership proof π_u^m for any auction* AucID*.*

Proof. An honest user u registers by choosing ρ_u and computing C_u, which then gets added in the deposits variable of $\mathcal{DC}$ (see REGISTERDEPOSIT() in Algorithm 2). The deposit (A_u, C_u) can only be removed from deposits through function MAKEINACTIVE(), SLASH() or WITHDRAW() (see Algorithm 2).

For an honest u, no honest IL Proposer will call MAKEINACTIVE(A_u), because u reveals its bid commitments or participates in all auctions using a NOP bid. A malicious IL Proposer *can* call MAKEINACTIVE(A_u) against honest u, however, this gives the honest user (or any other honest IL Proposer) a n_A window in which it can use UPDATEACTIVITY() to avoid becoming inactive.

Moreover, SLASH() or WITHDRAW() can only be called if either the user uses a h_u twice or withdraws its own funds, which the honest does not do.

Hence, deposits (A_u, C_u) of honest u will remain in the deposits. This means that u can create the path variable from root_s (the root of the merkle tree that contains all deposits in deposits) to (A_u, C_u) for some slot s (see ELGPRV() in Algorithm 3). Moreover, u can create the handle $h_u = \textsc{Hash}(\rho_u \parallel \mathsf{AucID})$ for any auction AucID and commitment $\mathsf{cm} = \textsc{Hash}(tx \parallel C_u)$ for its transaction tx. Hence, u can compute a valid witness $(\rho_u, \mathsf{A}_u, \mathsf{path}, tx)$ for the statement $(\mathsf{root}_s, h_u, \mathsf{cm})$ and, from the completeness property of the zero-knowledge proof system, a verifying membership proof π_u^{m} for any auction AucID. □

Lemma B.3 (Max Inactivity)**.** *A user can get a timestamp certificate for at most $n_c = 3n_A - 1$ auctions without revealing the corresponding bids.*

Proof. Let a user's last revealed auction id be AucID_ℓ. If the user does not participate for n_A auctions, at least honest IL Proposer will call MAKEOFFLINE(). This starts a n_A-auction countdown, for which the user remains active (and thus can get timestamp certificates). When the last of these auctions is about to end, there can be at most $n_A - 1$ more auctions started and still be running. A valid certificate can be generated for all of these $3n_A - 1$ auctions; however, as soon as the last auction ends, the user's deposit becomes

inactive and no other h_u can be generated with a valid π^{m}_u. □

Lemma B.4 (Uniqueness of handle per deposit)**.** *For each deposit (as indicated by C_u) and auction* AucID*, the user can create at most one handle h_u, such that π^{m}_u is a verifying membership proof for the statement* $(\mathsf{root}_s, h_u, \mathsf{cm})$.

Proof. Assuming that HASH() is a collision-resistant hash function, a user u cannot come up with $\rho'_u \neq \rho_u$, such that $\textsc{Hash}(\rho'_u) = C'_u$ and $C'_u = C_u$. Hence, u would have to use ρ' to create a valid membership proof π^{m}_u for a statement $(\mathsf{root}_s, h'_u, \mathsf{cm}')$, where $h'_u = \textsc{Hash}(\rho'_u \parallel \mathsf{AucID})$ and $\mathsf{cm}' = \textsc{Hash}(tx' \parallel C'_u)$, for an entry (A_u, C'_u) that does not exist in the merkle tree root_s. This is impossible, except with negligible probability, because it would contradict the collision resistance of HASH(), used to implement the Merkle tree. □

B.2. Proofs for the Entire Construction

Theorem 4.1 (Censorship Resistance)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time t_{end}, and has some pre-defined settlement slot S. The auction satisfies $\delta-$*SST Censorship Resistance with $\delta = (\Delta_g + \Delta)$ *(according to Definition 2).*

Proof. For every honest user $u \in \mathcal{U}$, let tx be a bid transaction created by u for auction AucID and sent to timestampers before clock time $t_{\text{end}} - \delta$ (i.e., $t_u \leq t_{\text{end}} - (\Delta_g + \Delta)$). From Lemma B.2, the honest user can always create such a bid.

By Lemma B.1, *User Drift Bound*, we know that

$$\max(\mathcal{T}_{\mathsf{hon}}) \leq t_u + \Delta + \Delta_g.$$

Also from Lemma B.1, *Honest Span*, we get that

$$\tau \leq \max(\mathcal{T}_{\mathsf{hon}}) \leq t_u + \Delta + \Delta_g \leq t_{\text{end}},$$

i.e., the median timestamp obtained has value at most t_{end}. Thus, if such a bid transaction with timestamp is sent to $\mathcal{P}_{il}$, it is still considered valid. Then, tx is included in the input bid set of auction AucID used by all honest IL Proposers when computing the auction outcome. □

Theorem 4.2 ((Relaxed) Hiding)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time t_{end}. Let* $\textsc{AucSet}(\tau)$ *represent the ongoing parallel auctions at clock time τ. Let $\mathcal{U} = \{\mathsf{A}_u : (\mathsf{A}_u, \_) \in \mathsf{deposits}\}$ represent the users in* deposits *in the slot right before clock time τ. The auction satisfies the following items from the* Relaxed t_h-Strong Hiding *property (Definition 3): (i) Indistinguishability (ii) Existential Obfusction within* $\textsc{AucSet}(\tau)$ *(iii) User Obfuscation within $\mathcal{U}$.*

Proof. Indistinguishability. Let us consider, towards a contradiction, that an auction protocol violates indistinguishability. We will show that using this auction's adversary as a subroutine, we can design an adversary that has an advantage in learning a pre-image of a hash function or learning witness of a zero knowledge proof.

Concretely, $\mathcal{A}$ outputs two transactions (tx_0, tx_1) and is given a challenge commitment $\mathsf{cm}_b = \textsc{Hash}(tx_b \parallel C_u)$ together with a ZK membership proof created as

$$\pi^{\mathsf{m}}_{u,b} \leftarrow \textsc{ZkProve}((\mathsf{root}_s, h_u, \mathsf{cm}_b), (\rho_u, \mathsf{A}_u, \mathsf{path}, tx_b)),$$

where $b \leftarrow \{0,1\}$ is drawn uniformly at random. Before time t_h, all information available to $\mathcal{A}$ consists of:

- the handle $h_u = \textsc{Hash}(\rho_u \parallel \mathsf{AucID})$,
- the commitment cm_b,
- the proof $\pi^{\mathsf{m}}_{u,b}$, and
- the timestamp certificate (τ, σ),

none of which reveals tx_b directly. i) Since h_u is independent of tx_b, there is no advantage that the adversary gains from this information. By hash function's pre-image resistance, ρ_u and AucID remain hidden with no advantage more than random guessing. ii) the timestamp certificate (τ, σ) contains only timestamps and signatures. Again no information about tx_b is available here.

By assumption, the adversary outputs a guess b' such that

$$\left|\Pr[b' = b] - \tfrac{1}{2}\right| \geq \epsilon(\lambda)$$

for some non-neglibile function ϵ.

Now, for the commitment $\mathsf{cm}_b = \textsc{Hash}(tx_b \parallel C_u)$, consider an adversary $\mathcal{A}_1(\mathcal{A})$, such that, when $\mathcal{A}$ outputs two messages (tx_0, tx_1), the adversary $\mathcal{A}_1$ outputs $(m_0 = (tx_0 \parallel C_u), m_1 = (tx_1 \parallel C_u))$. The challenger to the pre-image resistance function returns $\textsc{Hash}(m_b)$ and this value is passed to $\mathcal{A}$. When $\mathcal{A}$ outputs b, $\mathcal{A}_1$ outputs the same b. Since $\mathcal{A}_1$ must not have any advantage (by definition of HASH() being pre-image resistant), $\mathcal{A}$ can also not obtain any advantage through the commitment cm_b.

Lastly, for the proof $\pi^{\mathsf{m}}_{u,b}$, consider an adversary $\mathcal{A}_2(\mathcal{A})$. When $\mathcal{A}$ outputs two messages (tx_0, tx_1), adversary $\mathcal{A}_2$ chooses a random $b \leftarrow \{0,1\}$ and constructs the corresponding commitment

$$\mathsf{cm}_b = \textsc{Hash}(tx_b \parallel C_u).$$

It then submits the ZK statement $(\mathsf{root}_s, h_u, \mathsf{cm}_b)$ to its ZK challenger, receiving either a real or simulated proof π^* depending on the hidden bit β. $\mathcal{A}_2$ now continues the protocol with $\pi^{\mathsf{m}}_{u,b} = \pi^*$ and sends to $\mathcal{A}$ the tuple

$$(h_u, \mathsf{cm}_b, \pi^{\mathsf{m}}_{u,b}, (\tau, \sigma)).$$

Adversary $\mathcal{A}$ outputs b'. If $b = b'$, $\mathcal{A}_2$ outputs $\beta =$ simulated. Otherwise, output $\beta =$ real. However, since $\mathcal{A}_2$ must have no advantage in the zero-knowledge game, $\mathcal{A}$ cannot have an advantage in the indistinguishability game. □

Existential Obfuscation within $\textsc{AucSet}(\tau)$. Let us consider, towards a contradiction, that this advantage is not neglible. Before time t_h, all information available to $\mathcal{A}$ consists of:

- the handle $h^b_u = \textsc{Hash}(\rho_u \parallel \mathsf{AucID}_b)$,
- the commitment $\mathsf{cm} = \textsc{Hash}(tx \parallel C_u)$,
- the proof $\pi^{\mathsf{m}}_{u,b}$, and
- the timestamp certificate (τ, σ),

The proof follows the same arguments as *indistinguishability*. Only information $h_u^b = \textsc{Hash}(\rho_u \parallel \mathsf{AucID}_b)$ and the proof $\pi^{\mathsf{m}}_{u,b}$ are dependent on the auction ID. The timestamping-certificate time τ only reveals that $\mathsf{AucID}_b \in \textsc{AucSet}(\tau)$ and thus the adversary $\mathcal{A}$ can be used to build adversaries that have an advantage in learning a pre-image of a hash function or learning witness of a zero knowledge proof. □

User obfuscation within $\mathcal{U}$. Before time t_h, all information available to $\mathcal{A}$ consists of:

- the handle $h_u^b = \textsc{Hash}(\rho_b \parallel \mathsf{AucID})$,
- the commitment $\mathsf{cm}_b = \textsc{Hash}(tx \parallel C_b)$,
- the proof $\pi^{\mathsf{m}}_{u,b}$, and
- the timestamp certificate (τ, σ),

Here, the timestamp certificate (τ, σ) reveals only that whoever submitted the transaction is in $\mathcal{U}$. The other three elements are all similar to previous proofs: if advantage is obtained from either the handle h_u^b or cm_b, then we can make an adversary with an advantage in pre-image of a hash function, and if the advantage is obtained from proof $\pi^{\mathsf{m}}_{u,b}$, then we can make an adversary with an advantage in learning witness of a zero knowledge proof. □

Theorem 4.3 (Auction Participation Efficiency)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time t_{end}, and has some pre-defined settlement slot S. The cost to an honest active user if it does not win an auction tends to* 0*.*

Informal Proof. If the user is active and honest, then after depositing, the user never needs to make an on-chain call. This can be done by revealing its bids for auctions it participates in, and by submitting a proof of non-participation immediately if it does not. Even if a malicious IL Proposer calls MAKEINACTIVE(), an honest IL proposer with its history will always call UPDATEACTIVITY() before the user is ever made inactive. Thus, the only time the user pays a fee is when it registers its deposit, and thus the amortized cost over many auctions that the user participates but does not win in, will tend to 0. □

Theorem 4.4 (No Free Bid Withdrawal)**.** *Assume an auction that spans Algorithms 1 to 4, concludes at time t_{end}, and has some pre-defined settlement slot S. If* $\mathsf{minDeposit} \geq (3n_A - 1) \cdot u$*, meaning that, by not revealing a commited bid, the adversary pays an amount of at least u per auction, the auction satisfies the* No Free Bid Withdrawal *property.*

Proof. Consider that a bidder decides not to reveal a bid tx after receiving a timestamp certificate for some auction id AucID. The handle for this auction is unique, as proved in Lemma B.4, and was used to get the timestamp for tx. Thus, creating a proof of non-participation for h_u would double-use h_u, causing SLASH() to be called on-chain. Thus, this would imply that the user can never generate a proof of non-participation for such an AucID. Thus, it can never withdraw its deposit from the contract loosing its deposit.

However, the bidder can do so for at most $n_c = 3n_A - 1$ auctions (from Lemma B.3) and thus the amortized cost for not revealing any valid winning bid is at least $\frac{\mathsf{minDeposit}}{n_c}$, which is at least u, per the theorem statement. □