Fast and Gas-efficient Private Sealed-bid Auctions

JONAS BALLWEG, Hong Kong University of Science and Technology, Hong Kong
AMIR KAFSHDAR GOHARSHADY, University of Oxford, United Kingdom
ZHAORUN LIN, Hong Kong University of Science and Technology, Hong Kong

We consider the classical problem of running a decentralized and trustless auction, using a smart contract, on a programmable blockchain such as Ethereum. In our setting, there are n bidders who have paid a deposit to join the protocol. Each bidder i can make a bid $1 \leq b_i \leq m$ and our goal is to find the highest bid ($\max_i b_i$) and its corresponding bidder ($\text{argmax}_i b_i$) in a publicly-verifiable manner. Each bidder must be unaware of others' bids when making their own and should not be able to change their bid after having committed to it. Additionally, and most importantly, we aim to provide privacy to the losing bidders, ensuring that their bids remain undisclosed. This is particularly crucial in use-cases with repeated auctions in which knowledge of the bids in the previous auctions can affect the bidders' strategies. Formally, the information gained by any observer, whether a participant in the protocol or not, should precisely consist of the winning bid and its bidder and nothing more.

Given that we work in a blockchain setting, there are two natural and distinct measures of performance: (i) running time, i.e. the number of blocks required for the auction to terminate, and (ii) gas usage, i.e. the computation cost paid as transaction fees by each bidder. Previous methods either leak everyone's bids (publicly or to a centralized auctioneer) or have prohibitive time complexity or gas usage. For example, a classical commit-reveal scheme runs in $O(1)$ time and costs $O(1)$ gas per bidder, but leaks everyone's bids publicly. A naïve Dutch auction achieves $O(1)$ gas usage per bidder, while requiring an untenable $\Omega(m)$ time. It keeps losing bids confidential, but fails to ensure that bidders have no information about others' bids when making their own. On the other hand, non-blockchain auction protocols and those based on multi-party computation do not distinguish between time and gas, leading to a prohibitive gas usage of $\Omega(n)$ for each bidder.

In this work, we present a novel yet simple protocol for private sealed-bid auctions on the blockchain. Our protocol is decentralized and trustless and provides the security guarantees mentioned above. It is also both time- and gas-efficient. Our approach takes $O(\log m)$ time and costs $O(\log m)$ units of gas for each bidder. It also guarantees observational determinism with respect to all losing bids.

1 Introduction and Related Works

Designing decentralized auction protocols is a classical and well-studied topic in distributed systems and especially in the context of blockchain [1, 4, 23]. In this work, we consider the problem of implementing a sealed-bid first-price auction as a smart contract on a programmable blockchain such as Ethereum or Cardano. Our goal is to design a protocol that is both fast, in terms of number of blocks needed to execute it, and gas-efficient, having a small cost for each participating bidder. We also aim to provide privacy guarantees to non-winning participants ensuring that their bids remain private and only the highest bid/bidder is revealed.

Transparent Auctions. An auction protocol is called *transparent* if every participant's bid is publicly revealed by the end of the protocol. A common approach in transparent auctions is the commit-reveal scheme, as implemented in several smart contract-based protocols such as [36]. In these schemes, bidders first commit to their bids and then reveal them during a designated phase. Although this method ensures trustless execution, it ultimately exposes all bid values, offering no bid privacy. Unlike traditional commit-reveal auctions, we aim to ensure privacy for the losing bidders, i.e. guarantee that no information is leaked about their bids.

Authors' Contact Information: Jonas Ballweg, jballweg@connect.ust.hk, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong; Amir Kafshdar Goharshady, amir.goharshady@cs.ox.ac.uk, University of Oxford, Oxford, United Kingdom; Zhaorun Lin, zlinba@connect.ust.hk, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong.

Anonymous Auctions. An auction protocol is *anonymous* if it leaks the bids but no one can infer their ownership and know which bid belongs to which bidder. Two prominent directions in designing anonymous auctions are based on ring signatures and blind signatures.

Ring Signatures. A ring signature is a cryptographic primitive that allows any member of a group to sign a message anonymously, making it infeasible to determine which member produced the signature. It has been proposed for building anonymous auction systems where the bid values can be seen by everyone after the auction ends but they are not tied to someone's identity. For example, [46] employs ring signatures in a blockchain setting to achieve bidder anonymity. However, the approach is not fully trustless since the auctioneer may later deanonymize the bidders. Similarly, [20] implements an anonymous first-price sealed-bid auction using ring signatures, relying on a centralized auctioneer with the potential to compromise anonymity. In [39], ring signatures are used to protect bid confidentiality and bidder identity, yet the auctioneer still retains the capability to reveal bid values.

Blind Signatures. Blind signatures [9] are digital signatures in which the message is first "blinded" so that the signer cannot see its content, ensuring that the signature cannot later be linked back to the original message. This method is used in [44], where blind signatures combined with time-lock encryption decouple bid values from bidder identities. Although the bid values become publicly verifiable after the auction ends, they are not tied to individual bidders.

Private Auctions. We say an auction is private if it does not leak any information about the losing bids. See Section 2 for a formal definition. Most private auctions in the literature are based on either homomorphic encryption or multiparty computation protocols. They are not designed for the blockchain setting and often require costly computations that, if implemented in a smart contract, would lead to an untenable gas usage of $\Omega(n)$ per participant, where n is the number of bidders.

Homomorphic Encryption. Homomorphic encryption is a cryptographic tool that allows users to compute directly on encrypted data without having to decrypt it first. It naturally fits well in auction protocols, as bidders may securely perform calculations on their encrypted bids [3, 25]. For example, a Pedersen commitment [35], which is a type of homomorphic commitment scheme, is deployed in [42] and [17]. In the Riggs protocol [42], each bidder has a balance (commitment) recorded in the auction house, which represents the total amount each bidder may use to place bids in the auctions being hosted. A Pedersen commitment is particularly useful in this case as bidders can directly update their balances or place a bid without revealing the amounts.

Multiparty Computation. Many auction protocols such as [16, 28, 32, 45, 49] utilize secure multiparty computation to secretly compute the result of an auction while keeping the bids private. For example, Cryptobazaar [32] is a protocol that runs in $O(n)$ time and can be generalized to an ith-price auction that discloses only the ith highest bid and nothing else. It uses the Anonymous Veto protocol of [19] to blind the bids from the auctioneer and everyone else.

Further Related Works. Several works have examined transparent auctions on blockchain platforms. In transparent auctions, bidders and bid values are publicly known. For instance, [27, 36, 41] explore English, Dutch, and sealed-bid auctions and challenges such as front-running, collusion, and inefficiencies in bid revelation. Other studies provide comprehensive overviews of blockchain-based auction approaches [41], practical implementation insights [33], and discussions on protocols that rely on broadcast or multiparty channels [11, 31]. Additional research has focused on optimizing sealed-bid auctions for liveness [21], or developing collusion-resistant systems [43].

Our Contribution. We provide a novel and simple auction protocol that can be implemented as a smart contract and combines ideas from Dutch auctions, commit-reveal schemes and binary

interval trees. Our protocol takes $O(\lg m)$ time where m is the maximum allowed bid. Similarly, every bidder in the protocol pays for $O(\lg m)$ units of gas in the worst case. Our protocol is decentralized, premissionless, and trustless. It also guarantees both bid independence, i.e. every bidder is unaware of others' bids when making their own, and privacy for losing bidders, ensuring that only the highest bid and its corresponding bidder are publicly identified. Finally, we guarantee that the results are publicly verifiable.

Organization. Section 2 explains our setting and formalizes the problem. This is then followed by our protocols. In Section 3 we start with a simple combination of commitment schemes and Dutch auctions which provides the desired privacy guarantees and requires $\Theta(m)$ time but only $O(1)$ gas for each bidder. In Section 4, we introduce the *binary auction tree* data structure and use it to design a protocol with $O(\lg m)$ time and $O(\lg m)$ gas cost for each bidder. This is followed by an extension in Section 5 which ensures the binary auction tree does not leak information about non-winning bids.

2 Preliminaries and Problem Statement

In this section, we first outline our setting and assumptions about the blockchain and smart contract environment. These are standard assumptions and can be skipped by readers who are already familiar with this setting. We then define our auction problem and the desired security guarantees.

Programmable Blockchain. Our goal is to design a protocol that can be implemented as a smart contract on common programmable blockchains such as Ethereum [5] or Cardano [7, 29]. Our protocol shall not be limited to a particular architecture such as the accounting model of Ethereum or the extended UTXO employed by Cardano. Thus, in our setting, a smart contract is simply an immutable program which can own and transfer money and has functions that can be called by anyone on the network. The underlying blockchain layer ensures that all users reach a consensus on the sequence of function calls and thus the state of the smart contract. We consider the consensus layer as a black box and do not analyze its details, e.g. whether it uses proof-of-work or proof-of-stake. All participants have access to the same consensus and can see the state of the blockchain and smart contracts. The transactions are divided into a number of blocks, each corresponding to a particular time period. If several transactions are made during a single block's time period, their order in the block is non-deterministic and arbitrary. We also assume that there is no limit on the number of transactions per block and no transactions are "dropped" by the miners. In other words, all valid transactions reach the smart contract. In practice, this can be ensured by allowing several blocks' time for each transaction to be mined, i.e. scaling up the time limits by a constant factor. We assume that the smart contract has access to block numbers. Finally, the runtime of a protocol is the number of blocks from its inception to completion.

Gas. In Bitcoin, every transaction includes a fee that is paid to the miner who adds it to the consensus chain [30]. In programmable blockchains, the same mechanism is also used to avoid denial-of-service attacks. Every transaction added to a block in the consensus chain has to be executed by everyone on the network. Thus, the transaction's originator, i.e. the person calling the function, has to pay a transaction fee which is proportional to the computational resources it uses. This payment will dissuade users from invoking costly executions that may clog the network. The execution cost is called *gas* and is usually obtained by a fixed formula that assigns costs to every atomic operation in the smart contract. This is hard-coded in the blockchain protocol. For example, in Ethereum, the beigepaper contains a table of gas costs [14]. Gas fees are significant and cost Ethereum users almost 4 billion USD per year [15]. Thus, when designing a blockchain-based protocol, we must distinguish between off-chain computation, i.e. computation done on the user's

own machine which does not incur gas costs, and on-chain computations, i.e. calls to smart contract functions which cause computations performed by everyone on the network and do incur gas fees.

Deposits. Distributed auction protocols often include mechanisms to detect cheating by participants. In a blockchain setting, smart contracts can receive and own money in the form of an underlying cryptocurrency. Thus, we can additionally assume that the participants are required to sign up with the smart contract and pay a deposit to take part in the protocol. This way, any detection of dishonest behavior can immediately be punished by confiscating their deposits.

Identities. Blockchain environments use asymmetric cryptography and identify users by their public keys. All transactions, including all function calls to smart contracts, have to be signed by the originator. The environment is pseudonymous in the sense that a user can create as many identities as they wish by simply generating more secret/public key pairs. We will exploit this property in our protocol, where users can generate a new identity to send a message to the smart contract without it being connected to their previous identity. In practice, users can have many identities before taking part in the protocol and can use mixing services to fund all of their accounts (public keys) without disclosing their connection to the same person [2, 34].

Bidders. We consider an auction with n bidders numbered from 1 to n. We use pk_i to denote the public key (identity) of the i-th bidder and sk_i to denote its corresponding secret key. Naturally, pk_i is public knowledge whereas sk_i is only known to i.

Auction Problem Statement. Our goal is to design a sealed-bid auction that can be implemented as a smart contract. Each bidder i should be able to place a bid b_i by an interaction with the auction smart contract. We assume there is a known global maximum bid m and every b_i is between 1 and m. The contract should then obtain the maximum bid $(\max_i b_i)$ and its bidder $(\operatorname{argmax}_i b_i)$[†]. These values must be obtained in a publicly-verifiable manner, i.e. anyone with access to the blockchain should be able to verify that the auction has completed successfully and the highest bid/bidder are identified correctly.

Security Guarantees. We require our auction to satisfy the following security properties. The first three are classical and can be obtained even by the simplest commit-reveal schemes. Thus, we will mainly focus on the fourth property below:

(1) ***Decentralization and Permissionlessness.*** Anyone on the blockchain network can sign up to take part in the auction and no party has permissions to perform operations that are not allowed to any other party.
(2) ***Trustlessness.*** No party is assumed to be honest. If a party is not following the protocol correctly, this should be identified and punished.
(3) ***Bid Independence.*** No bidder should be able to change their own bid after learning any information about other bids. This is also called the *sealed-bid* property.
(4) ***Privacy for Losing Bidders.*** If a bidder i is not the highest bidder, then no information should be leaked about b_i. Note that this property is violated even if b_i or some information about b_i is leaked without it being directly connected to i. For example, if an observer realizes that one of the bids was a particular value, without knowing who the bid belonged to, we still consider this a breach of privacy.

Observational Determinism. We formalize the fourth property (privacy for losing bidders) above using the notion of observational determinism which is standard in the computer security literature and often used in the context of concurrent programs [12, 22, 26, 37, 47]. Let j be an

[†]If the sequences have several maximal elements, we assume that argmax returns the set of indices in which the maximum value appears. The protocols are explained as if the auction's winner is unique, but it is trivial to extend them to cases with several winners.

observer, i.e. a user with access to the blockchain network who may or may not be one of the bidders. In one run of the protocol, the observation made by j is the sequence of all transactions and blocks j sees on the blockchain network, which may originate from any participant or miner, together with the timestamps at which they see such transactions/blocks. We denote one such observation with o. Let $B = \langle b_1, b_2, \ldots, b_n \rangle$ be the sequence of bids. We say o is consistent with B from j's point-of-view and write $B \models_j o$ if it is possible that j observes o when the bidders' bids are according to B. Intuitively, if $B \models_j o$ and $B' \models_j o$, when j observes o they cannot distinguish whether the bids were according to B or B'. We say two bid sequences $B = \langle b_1, b_2, \ldots, b_n \rangle$ and $B' = \langle b'_1, b'_2, \ldots, b'_n \rangle$ are compatible and write $B \rightleftharpoons B'$ if $\max_i b_i = \max_i b'_i$ and $\operatorname{argmax}_i b_i = \operatorname{argmax}_i b'_i$.

Based on the definitions above, a protocol provides *privacy for losing bidders* if we have:

- For every observer j who is not a bidder, if an observation o is consistent with B, then it is consistent with any B' that is compatible with B. Formally,

$$\forall o \; \forall B \; \forall B' \;\; \left(B \models_j o \;\wedge\; B \rightleftharpoons B' \right) \Rightarrow B' \models_j o.$$

- The same property should hold for every *bidder* j except that the bidder knows their own bid b_j. Thus, if j is a bidder and an observation o is consistent with B from their point-of-view, then it should be consistent with any $B' \rightleftharpoons B$ as long as $b'_j = b_j$. Formally,

$$\forall o \; \forall B = \langle b_1, b_2, \ldots, b_n \rangle \; \forall B' = \langle b'_1, b'_2, \ldots, b'_n \rangle$$

$$\left(b_j = b'_j \;\wedge\; B \models_j o \;\wedge\; B \rightleftharpoons B' \right) \Rightarrow B' \models_j o.$$

  A bidder might take part in the auction with several identities and make several bids. In such cases, we should extend the definition above accordingly to require that the observations match on all bids made by the same person. This also models collusions between bidders.

The formal definition above precisely captures our desired privacy concept. Every observer or colluding set of observers would only be able to distinguish between B and B' if they can do so using their own bid(s) and the information about the maximum bid/bidder. Thus, the observation does not leak any information about the losing bids/bidders.

Efficiency Metrics. To analyze the efficiency of our protocol, we consider the runtime and the maximum gas usage of any bidder. Recall that the runtime is the number of blocks required to execute the protocol. This is not the same as our protocol's computational complexity, given that a single block may contain several transactions/function calls. On the other hand, the gas usage is a closer concept to computational complexity. When a user invokes a function with computational complexity f, it will cost them $\Theta(f)$ in gas. Given that smart contract functions can be called by different users, this cost might be divided among them based on the protocol's requirements. We consider the maximum cost paid by a single user/bidder.

3 A Dutch Auction with Commitments

Our first protocol is a combination of classical commitment-scheme auctions and Dutch auctions. It provides excellent efficiency in terms of gas, requiring only $O(1)$ gas usage for each bidder. Note that this is asymptotically optimal since each bidder must at least sign up with the protocol. However, it requires $\Theta(m)$ time where m is the maximum possible bid.

Protocol 0. Dutch Auction. In a Dutch auction, an auctioneer starts with a high asking price m and keeps lowering the price until one of the bidders agrees to pay it [13]. This kind of auction originates in Dutch flower markets and can be easily implemented as a smart contact consisting of the following steps:

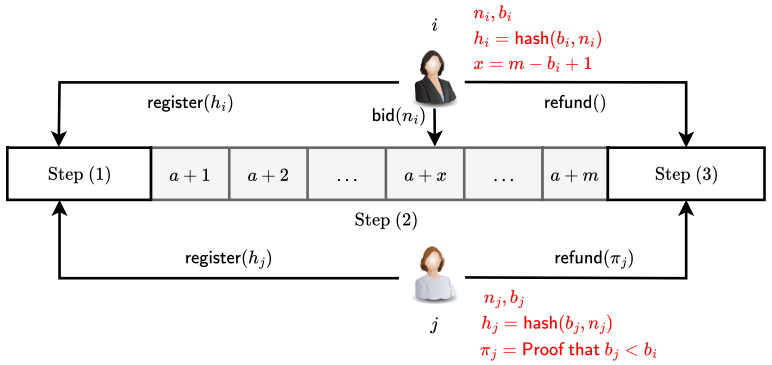

Fig. 1. Timeline of function calls by a winning bidder i and a losing bidder j to the auction smart contract in Protocol 1. In this figure, Step (1) ends at block number a and Step (2) spans blocks $a+1$ to $a+m$. Red values are private computations made by the bidders on their own machines.

(0) ***Parameter Setup.*** The organizer deploys the auction on the blockchain and chooses the deadlines, in terms of block numbers, for each of the following steps. For each step i, the organizer fixes two block numbers $[\tau_i, \tau_i']$ and the contract accepts function calls of step i only in this period. The organizer also sets a parameter d, which is the deposit each bidder must pay to join the auction and m which is the maximum allowed bid. This step is the same for all of our future auction protocols and thus we omit it for brevity. Some protocols need additional parameters whose values will also be set in this step.

(1) ***Registration.*** Anyone on the blockchain network can call a REGISTER() function in this step, paying a deposit d. The contract keeps track of the public key pk_i of every registering bidder.

(2) ***Countdown.*** This step lasts for exactly m blocks. Throughout this step, any registered bidder can call a function BID() in the contract. The bidder does not need to provide the value of their bid. In the x-th of the m blocks, the contract only accepts bids of value $b_i = m - x + 1$. Thus, the time of the bid uniquely determines its value*. The auction concludes as soon as a bid is made. The first bid is automatically the highest bid and its bidder is the auction's winner.

(3) ***Refund.*** In this step, all bidders can call a REFUND() function to receive their deposit d back. Based on the particular use-case, the winner might be excluded.

Although the simple protocol above provides privacy for losing bidders, whose bids are never revealed, it does not guarantee bid independence. Indeed, when at time x the highest bidder i decides to bid $b_i = m - x + 1$, this is done with the knowledge that no one else's bid is higher than the value b_i. This violates bid independence. For example, a bidder who intends to bid 100 USD for a batch of Dutch flowers might change their bid to 95 USD when they realize that no one else has bid 101 USD or more. To fix this, we combine the classical commit-reveal auction model with the above protocol.

Protocol 1. Dutch Auction with Commitments. Our new protocol consists of the following steps:

*To avoid issues due to network latency, one can set a longer period of several blocks for each bid value, thus scaling the time by a constant factor.

(1) ***Registration and Commitment.*** Anyone on the blockchain network can register. To do so, they must first commit to their bid b_i. They choose a random nonce n_i and compute $h_i = \text{HASH}(b_i, n_i)$. Here, HASH is a cryptographic hash function. They then call $\text{REGISTER}(h_i)$ and pay a deposit d. The contract saves the registrant's public key pk_i and their declared hash h_i.
(2) ***Countdown.*** This step lasts for exactly m blocks and is similar to the previous protocol. Throughout this step, any registered bidder i can call a function $\text{BID}(n_i)$ in the contract. The bidder does not need to provide the value of their bid, but only their random nonce n_i. In the x-th of the m blocks, the contract only accepts bids of value $b_i = m - x + 1$. When $\text{BID}(n_i)$ is called by bidder i, the contract verifies that $h_i = \text{HASH}(b_i, n_i)$. If not, the bidder will be penalized and the function call ignored. As before, the first bid that passes the hash check is automatically the highest bid.
(3) ***Refund.*** In this step, each bidder j can call a function $\text{REFUND}(\pi_j)$ to receive their deposit d back. The winner might be excluded based on the use-case. Moreover, any non-winning bidder must provide a proof π_j showing that their bid b_j was smaller than the winning bid b_i. For this, one does not need to publish the nonce n_j and can use any standard zkSNARK such as Groth16 [18] instead. More specifically, π_j is a proof that j knows values n_j and b_j such that $b_j < b_i$ and $\text{HASH}(b_j, n_j) = h_j$. The contract verifies π_j and issues the refund only if π_j is valid. Otherwise, the bidder's deposit will be burned.

Efficiency. Our protocol above has the same runtime and gas usage as a vanilla Dutch auction. The time is dominated by the countdown which takes $\Theta(m)$ blocks in the worst case. On the other hand, each bidder pays only $O(1)$ in gas, since they have to send a single constant-sized registration transaction in Step (1), followed by a single bid in Step (2) only if they are the winner, and a single refund transaction in Step (3) if they are not the winner, which verifies a constant-sized zkSNARK with constant gas usage.

Security Analysis. Public verifiability is immediate since the contract verifies everything and anyone on the blockchain network can simply run it, too. Decentralization and trustlessness are easy to check. Bid independence is achieved since every bidder commits to their bid in Step (1) and thus cannot change it later. At this point, they have no information about other bids. Privacy for losing bidders is obtained by design since no losing bid is revealed in Steps (2) and (3) and each losing bidder simply provides a zkSNARK certifying they have not won the auction.

4 Binary Auction Trees

While Protocol 1 of the previous section has all the desired security properties, it takes $\Theta(m)$ blocks to execute. This is prohibitively large for real-world auctions. For example, if we have an auction in which $m = 10^6$ and each block takes 13 seconds, as it does on Ethereum, then the countdown step would require almost 150 days. In this section, we provide a protocol that improves the runtime to $\Theta(\log m)$ blocks, albeit at a slightly increased cost of $\Theta(\log m)$ gas for a bidder in the worst case.

Binary Auction Tree. The idea is to use a binary interval tree of possible bids, which we call a *binary auction tree* (BAT). The root of a BAT corresponds to the interval $[1, m]$. Each vertex v of the tree that is labeled by the interval $[x, y]$ will have two children, the left one corresponding to $\left[x, \lfloor \frac{x+y}{2} \rfloor\right]$ and the right being labeled by the interval $\left[\lfloor \frac{x+y}{2} \rfloor + 1, y\right]$. Each leaf will correspond to a single possible bid value, i.e. an interval $[x, x]$. For example, Figure 2 shows the BAT for $m = 15$.

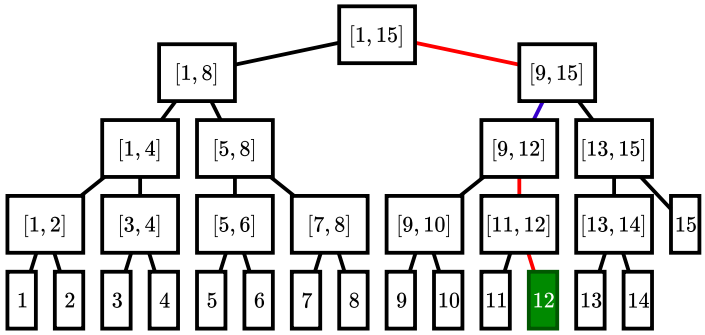

Fig. 2. A Binary Auction Tree (BAT) in which $m = 15$ and the path taken if the maximum bid is 12. The red edges correspond to explicit calls to RIGHT() and the blue edge is an implicit move to the left child.

Intuition. In our second protocol, the smart contract starts at the root of the BAT and keeps traversing it down until it reaches a leaf. At each level of the tree, the bidders must collectively guide the contract towards the highest bid. Since each bidder knows their own bid, they should send a message to the contract if their bid is in the right child of the current node. If the contract does not receive any such message by a particular deadline, it moves to the left child. Otherwise, it moves to the right child. This continues until the contract finds the maximum bid. The actual protocol is a bit more involved as (i) a smart contract can change states only when one of its functions is called, i.e. it cannot automatically invoke itself, and (ii) we need to penalize dishonest bidders.

Protocol 2. Auction using a BAT. Our protocol consists of the following steps:

(1) ***Registration and Commitment.*** Same as in Protocol 1.
(2) ***Path Finding.*** Suppose the BAT has depth k, i.e. the distance from the root to the farthest leaf is k edges. This step will take exactly k blocks time. If the previous step ends at block number a, then the current step runs from block $a + 1$ to $a + k$[†]. The smart contract implicitly keeps track of its position in the BAT by saving the interval corresponding to the current node, as well the number of steps taken so far. In the time period of block $a + t$, if the smart contract is in a non-leaf vertex with corresponding interval $[x, y]$, then in the next step it should go to either the left child $\left[x, \lfloor \frac{x+y}{2} \rfloor\right]$ or the right child $\left[\lfloor \frac{x+y}{2} \rfloor + 1, y\right]$. If a bidder i realizes that their bid b_i belongs to the right child, i.e. $\lfloor \frac{x+y}{2} \rfloor + 1 \leq b_i \leq y$, they should call the function RIGHT() of the smart contract. This call tells the contract to move to the right child. Importantly, since we do not want bidder i to publicly disclose that their bid b_i is in a particular interval, this call is not performed using the identity pk_i that was used for registering bidder i in Step (1), but instead using several pseudonyms, i.e. different identities generated by the same bidder only for this call. The number of such pseudonymous calls is randomly chosen by the caller. Moreover, every call to RIGHT() must have a deposit d' attached to it. The purpose of this deposit will soon become clear. If a RIGHT() transaction is already issued by someone else in the current block, an honest bidder will not make the calls at all. Otherwise, they will make at least r calls to RIGHT(). r is a parameter fixed in

[†]As in the previous protocols, one can scale this to several blocks for each edge.

Step (0). The contract will ignore repeated calls to RIGHT() in the same block and return their deposits[‡].
A further implementation detail in this step is that smart contracts cannot generally invoke their own functions automatically. Thus, moving right is always by a function call from a bidder, but moving left is implicit and is only performed when the next right move is called or the time limit for Step (2) expires and a function in Step (3) is called. For example, a pseudocode for RIGHT() is as follows:

```
step ← 0
x ← 1
y ← m
procedure RIGHT
    t ← block.number − a
    while step < t − 1 ∧ x ≠ y do        ▷ Implicit left steps
        y ← ⌊(x+y)/2⌋
        step ← step +1
    if x ≠ y then                        ▷ Explicit right step
        x ← ⌊(x+y)/2⌋ + 1
        step ← step +1
```

Example. In Figure 2, if the highest bid is 12, then the highest bidder calls RIGHT() in the first block of Step (2) moving to $[9, 15]$. This call is done using a different identity (pseudonym) than the one used for registration. There might be other calls to RIGHT() at this point, too, but the contract will ignore repeated calls. Then, in the second block of Step (2), no one calls RIGHT(). Thus, all bidders realize that the contract has implicitly moved to $[9, 12]$ but this change is not yet executed in the contract. In the third block, the highest bidder calls RIGHT() again. At this point, the contract first performs the left move from the last block, going to $[9, 12]$ and then the current right move going to $[11, 12]$. Finally, in the fourth block, the highest bidder calls RIGHT() and the contract ends up in leaf 12.

(3) ***Revealing the Highest Bid.*** At the end of the previous step, we reach a leaf of the tree that corresponds to a particular bid value b_i belonging to a bidder i. In this step, the winner i must call BID(n_i) using their original identity pk_i and provide their nonce n_i. The contract first takes any remaining implicit left steps to find the value b_i, and then checks that HASH(b_i, n_i) = h_i. If no bidder calls BID(n_i) successfully within the time limit of this step, then the last person to call RIGHT() has been dishonest. In this case, anyone can call a function named BLAME(). The contract then confiscates the deposit d' of the last person who moved RIGHT(), goes back to immediately after the second-last call to RIGHT(), or the root if no such call exists, and redoes Step (2) to find a new leaf.

(4) ***Refund.*** This step is exactly the same as in Protocol 1. Each bidder j can call a function REFUND(π_j) to receive their deposit d back. Every non-winning bidder must provide a zkSNARK proof π_j showing that their bid b_j was smaller than the winning bid b_i. Additionally, all d' deposits for moving right can be refunded.

Efficiency. In this protocol, anyone can take part in guiding the contract through the binary auction tree. This is because we want to allow the bidders to use pseudonyms, i.e. identities other than the ones used in the registration phase, to guide the path. This does not affect our runtime. Note that spurious calls to RIGHT() are disincentivized. If they cause the contract to exceed the

[‡]In practice, the preferred design is to include the interval $[x, y]$ as a parameter and call RIGHT($[x, y]$) to avoid attacks that reuse stale function call transactions.

actual maximum bid, then they will be punished since each move to the right requires a deposit d'. If they do not exceed the maximum bid, they have no effect on the correct execution of the protocol. Thus, in the presence of rational parties, we can analyze the runtime and gas usage with the assumption that we have no such dishonest calls. In this case, the runtime bottleneck is Step (2) which requires $k = \Theta(\log m)$ blocks. Moreover, each bidder has to pay for gas used in one function call for registration, at most k calls to RIGHT() in Step (2) and then at most one call in each of Steps (3) and (4). Each call uses constant gas. Thus, the worst-case gas usage of a bidder is $\Theta(\log m)$.

Further Incentives. In the protocol above, the highest bidder is incentivized to correctly guide the contract to the leaf corresponding to their bid. Otherwise, they will lose their initial deposit d. However, at each step of the path, the protocol requires an honest bidder who wants to go right to submit not just one, but several calls to RIGHT() from different identities. To incentivize this, we can edit the smart contract to remember the first r calls to RIGHT() at each step and later pay a fixed reward to each of them. The reward will be taken from the bidders' initial deposits and its amount, as well as r, are parameters set in Step (0). This way, if several bidders know that we should go right at a particular step, they will compete on sending the information to the contract as soon as possible by creating many calls to RIGHT()[§].

Security Analysis. Public verifiability, decentralization, trustlessness and bid independence are achieved by arguments similar to the case of Protocol 1. Thus, we focus on analyzing privacy for losing bidders. Let j be a losing bidder. Given that j has signed up in the contract using the identity pk_j but made calls to RIGHT() by different identities, no one can observe anything about b_j that is connected back to j. This guarantee is sufficient for many real-world use-cases but is weaker than the formalism using observational determinism provided in Section 2. Indeed, this formalism is not satisfied by Protocol 2. As an example, consider the BAT in Figure 2 and suppose the highest bidder i is bidding $b_i = 12$. Suppose j is the second-highest bidder and $b_j = 10$. In the first step, i plans to call RIGHT() but observes that someone else calls RIGHT() first. This tells i that there is at least one other bid in the range $[9, 15]$. Although the identity of j is kept secret, the information that the second highest bid is also in $[9, 15]$ is enough to violate our strict privacy requirement. Of course, when i wins the auction, they will know that the second highest bid is in $[9, 12]$.

5 Fake Bids and Observational Determinism

While our Protocol 2 does not satisfy observational determinism as defined in Section 2, it comes quite close to it. There is no link between the right moves on the tree and the original identities of the bidders. Additionally, any leaked information is only about the second-highest bid. Intuitively, if the highest bidder plans to move right at some point but observes that someone else made the move first, they will know that the second-highest bid is in the right subtree, too. However, such an observation can be made irrespective of the other bids and thus does not leak any information about them. From the point-of-view of anyone other than the highest bidder, observational determinism is already satisfied. If an observer j who is not the highest bidder observes several calls to RIGHT() at a particular block, they cannot know if the calls originated from the same person who is using several pseudonyms or a number of different people. This was the reason behind issuing each RIGHT() calls many times. Thus, they only gain information about the highest bid, which is not a violation of our privacy requirements.

Protocol 3. Auction using a BAT and Fake Bids. To provide privacy for the second-highest bidder and ensure the desired observational determinism from the point-of-view of the highest

[§]Another implementation detail is that RIGHT() should not allow itself to be called from any other smart contract. Thus, every call to RIGHT() is in a separate transaction.

bidder, we simply allow f of the bidders to each make an additional fake bid. f is a parameter set in Step (0). Specifically, our Protocol 3 has the following steps:

(1) ***Registration and Commitment.*** Same as in Protocols 1 and 2.

(2.1) ***Selecting Fake Bidders.*** f of the n bidders are selected randomly to be *fake bidders*. To choose the fake bidders, the smart contract relies on the output of a blockchain-based random number generator such as Randao [48]. Random number generation is a well-studied topic in blockchain and there are many efficient, tamper-proof and secure solutions available [8, 38]. Specifically, if the output of the RNG service in the previous block is ρ and RAND is a pseudo-random number generator, then ρ is chosen as the seed and RAND is called f times to choose the f fake bidders. In implementation, RAND can be instantiated to any cryptographic hash function.

(2.2) ***Computing Fake Bids.*** If a bidder i is chosen as a fake bidder, they first generate a fake bid

$$b'_i = (\text{RAND}(\rho, n_i) \mod m) + 1. \tag{1}$$

Note that the fake bid is uniformly distributed and depends on the RNG output ρ and the bidder's nonce n_i. They then take part in the protocol with both original and fake bids, i.e. b_i and b'_i.

(2.3) ***Path Finding.*** This step is exactly the same as Step (2) of Protocol 2, except that the fake bidders call RIGHT() whenever either of their two bids is in the right subtree.

(3) ***Revealing the Highest Bid.*** At the end of the path-6finding step, we reach a leaf of the tree. If the leaf corresponds to a real bid b_i, then the bidder i must call BID(n_i). The contract verifies this as in the previous protocol. Otherwise, if no such bid is declared, the leaf corresponds to a fake bid b'_i. In this case, the fake bidder i must call FAKEBID(π'_i) and provide a zkSNARK π'_i proving that their fake bid b'_i is indeed the leaf we have ended up in. Given that ρ is public knowledge, π'_i will simply be a proof that i knows a value n_i which satisfies equation (1).

(4) ***Verification of Fake Bids.*** Every fake bidder j must call FAKEBID(b'_j, π'_j), providing their fake bid b'_j and a zkSNARK proof that it was computed according to equation (1). If b'_j is larger than the leaf reached in Step (3) or π'_j is invalid or not provided, the contract confiscates the deposit of j and disallows them from continuing to participate in the auction.

(5) ***Refund or Reset.*** At this step, anyone who has paid a deposit to call RIGHT() can take the deposit back. If the leaf identified in Step (3) corresponds to a real bid, then the refund step is triggered, which works exactly as in Protocol 2, i.e. every non-winning bidder j can call REFUND(π_j), providing a zkSNARK π_j proving that their bid was smaller than the maximum bid b_i identified in Step (3) and receiving their deposit back. However, if the leaf identified in Step (3) corresponds to a fake bid, i.e. if the maximum bid b'_i is fake, the smart contract sets $m \leftarrow b'_i - 1$, goes back to Step (2.1) and performs a new walk from the root of the binary auction tree to a leaf.

Efficiency. The runtime and gas usage of Protocol 3 are similar to Protocol 2, except that the root-to-leaf walks on the BAT may be repeated several times. Specifically, suppose m is the maximum allowed bid and b_i is the largest bid. m decreases in each round, but we ignore this for ease of analysis. The probability that no fake bids surpass b_i is at least $(b_i/m)^f$. Thus, the expected number of times the protocol has to traverse a root-to-leaf path is less than $(m/b_i)^f$. Therefore, the expected runtime of the protocol and the expected gas usage per bidder are both in $O\left((m/b_i)^f \cdot \log m\right)$. If both f and m/b_i are small constants, i.e. only a few fake bids are allowed and the maximum allowed bid m is chosen reasonably so that it does not exceed the real maximum bid by more than a constant factor, then the asymptotic performance matches that of Protocol 2, i.e. $O(\log m)$ runtime and $O(\log m)$ gas usage per bidder.

Security Analysis. We only need to prove observational determinism since the other desired properties are inherited from Protocol 2. We assume that the constant $f > 1$ is chosen in a way that no one person may control all f fake bidders. In practice, since taking part in the protocol is costly due to the deposit and gas payments, even a small f suffices. Consider an observer j who makes an observation o during one root-to-leaf round of Protocol 3. o may contain several pseudonymous calls to RIGHT(). Since these calls are pseudonymous, j is unable to connect any of them to another bidder i. Moreover, j cannot obtain information about any of the bid values, except their own value b_j if they are a bidder. This is because j cannot distinguish the calls to RIGHT() that are made due to a real bid from those that are due to a fake bid. More specifically, if the round ends at a leaf that is then revealed in Step (3) as a real bid b_i with $i \neq j$, then it is possible that all the calls to RIGHT() in the current path were invoked by i's pseudonyms. Thus, from j's perspective, the observation is consistent with any bid sequence in which the maximum element is b_i. Alternatively, if j is the maximum bidder and in Step (3) b_j is revealed as the maximum bid, then the observation o is still consistent with any bid sequence in which the maximum element is b_j. This is because there might be a different bidder i whose fake bid is $b'_i = b_j$. This fake bidder would call RIGHT() according to the same path as j but will not reveal a real bid in Step (3). Finally, if the bid revealed in Step (3) is fake, the exact same argument establishes observational determinism, i.e. the observation is consistent with any sequence of bids whose maximum is less than the revealed fake bid of Step (3).

6 Conclusion

In this work, we presented a novel blockchain-based protocol for first-price sealed-bid auctions that guarantees privacy for losing bidders, i.e. that their bids are not leaked as formalized by the concept of observational determinism. Our protocol can be implemented as a smart contract on any programmable blockchain and is efficient in terms of both time and gas. It concludes within $O(\log m)$ blocks, where m is the maximum allowed bid, and each bidder pays an expected gas cost of $O(\log m)$. A limitation of our approach is that observational determinism models non-determinism in a system but does not consider probabilistic behavior or inference. Extending the auction protocol with a stronger probabilistic security guarantee, such as those provided by zero-knowledge protocols, is an interesting direction of future work.

References

[1] Ramiro Alvarez and Mehrdad Nojoumian. 2020. Comprehensive survey on privacy-preserving protocols for sealed-bid auctions. *Computers & Security* 88 (2020), 101502.
[2] Alireza Arbabi, Ardeshir Shojaeinasab, Behnam Bahrak, and Homayoun Najjaran. 2023. Mixing Solutions in Bitcoin and Ethereum Ecosystems: A Review and Tutorial. *arXiv preprint arXiv:2310.04899* (2023).
[3] Peter Bogetoft, Ivan Damgård, Thomas P. Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. 2006. A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation. In *FC*. 142–147.
[4] Chiara Braghin, Stelvio Cimato, Ernesto Damiani, and Michael Baronchelli. 2020. Designing Smart-Contract Based Auctions. In *SICBS*. 54–64.
[5] Vitalik Buterin et al. 2013. Ethereum white paper.
[6] Sayantan Chakraborty. 2021. Verifiable e-auction over a Block-Chain. *Dissertations - M Tech (CRS)* (2021).
[7] Manuel M. T. Chakravarty, James Chapman, Kenneth MacKenzie, Orestis Melkonian, Michael Peyton Jones, and Philip Wadler. 2020. The Extended UTXO Model. In *FC*. 525–539.
[8] Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Arash Pourdamghani. 2019. Probabilistic Smart Contracts: Secure Randomness on the Blockchain. In *IEEE ICBC*. 403–412.
[9] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *CRYPTO*. 199–203.
[10] Eric Chiquito, Ulf Bodin, and Olov Schelén. 2023. Survey on decentralized auctioning systems. *IEEE Access* 11 (2023), 51672–51688.
[11] Tarun Chitra, Matheus V. X. Ferreira, and Kshitij Kulkarni. 2023. Credible, Optimal Auctions via Public Broadcast. In *AFT*. 19:1–19:16.

[12] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal Logics for Hyperproperties. In *POST*. 265–284.

[13] Vicki M Coppinger, Vernon L Smith, and Jon A Titus. 1980. Incentives and behavior in English, Dutch and sealed-bid auctions. *Economic inquiry* 18, 1 (1980), 1–22.

[14] Micah Dameron. 2018. Beigepaper: an Ethereum technical specification. *Ethereum Project Beige Paper* (2018).

[15] EtherScan. 2024. Ethereum Daily Gas Used Chart. https://etherscan.io/chart/gasused

[16] Ehsan Ghasaei and Amirali Baniasadi. 2023. Blockchain-Based, Privacy-Preserving, First Price Sealed Bid Auction (FPSBA) Verifiable by Participants. In *BRAINS*. 1–8.

[17] Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. 2023. Cicada: A framework for private non-interactive on-chain auctions and voting. Cryptology ePrint Archive, Paper 2023/1473. https://eprint.iacr.org/2023/1473

[18] Jens Groth. 2016. On the Size of Pairing-Based Non-interactive Arguments. In *EUROCRYPT*. 305–326.

[19] Feng Hao and Piotr Zieliński. 2009. A 2-Round Anonymous Veto Protocol. In *Security Protocols*. 202–211.

[20] Ke Huang, Yi Mu, Fatemeh Rezaeibagha, Zheyuan He, and Xiaosong Zhang. 2021. BA2P: Bidirectional and Anonymous Auction Protocol with Dispute-Freeness. *Security and Communication Networks* 2021 (2021), 6690766.

[21] Maozhou Huang, Xiangyu Su, Mario Larangeira, and Keisuke Tanaka. 2024. Optimizing Liveness for Blockchain-Based Sealed-Bid Auctions in Rational Settings. Cryptology ePrint Archive, Paper 2024/1643. https://eprint.iacr.org/2024/1643

[22] Marieke Huisman, Pratik Worah, and Kim Sunesen. 2006. A Temporal Logic Characterisation of Observational Determinism. In *CSFW*. IEEE Computer Society, 3.

[23] Xuan Liu, Lu Liu, Yong Yuan, Yong-Hong Long, San-Xi Li, and Fei-Yue Wang. 2024. When Blockchain Meets Auction: A Comprehensive Survey. *IEEE Transactions on Computational Social Systems* 11 (2024), 4242–4254.

[24] Xuan Liu, Lu Liu, Yong Yuan, Yong-Hong Long, San-Xi Li, and Fei-Yue Wang. 2024. When blockchain meets auction: A comprehensive survey. *IEEE Transactions on Computational Social Systems* (2024).

[25] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. 2019. Homomorphic Time-Lock Puzzles and Applications. In *CRYPTO*. 620–649.

[26] John McLean. 1992. Proving Noninterference and Functional Correctness Using Traces. *J. Comput. Secur.* 1, 1 (1992), 37–58.

[27] Ciamac C. Moallemi and Dan Robinson. 2024. Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains. In *AFT*. 18:1–18:17.

[28] Jose A. Montenegro, Michael J. Fischer, Javier Lopez, and Rene Peralta. 2013. Secure sealed-bid online auctions using discreet cryptographic proofs. *Mathematical and Computer Modelling* 57 (2013), 2583–2595.

[29] Damian Nadales. 2023. A Formal Specification of the Cardano Ledger. https://github.com/input-output-hk/cardano-ledger/releases/latest/download/byron-ledger.pdf

[30] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto* (2008).

[31] Truc D. T. Nguyen and My T. Thai. 2021. A Blockchain-based Iterative Double Auction Protocol Using Multiparty State Channels. *TOIT* 21 (2021), 1–22.

[32] Andrija Novakovic, Alireza Kavousi, Kobi Gurkan, and Philipp Jovanovic. 2024. Cryptobazaar: Private Sealed-bid Auctions at Scale. Cryptology ePrint Archive, Paper 2024/1410. https://eprint.iacr.org/2024/1410

[33] Ilhaam A. Omar, Haya R. Hasan, Raja Jayaraman, Khaled Salah, and Mohammed Omar. 2021. Implementing decentralized auctions using blockchain smart contracts. *Technological Forecasting and Social Change* 168 (2021).

[34] Jaswant Pakki, Yan Shoshitaishvili, Ruoyu Wang, Tiffany Bao, and Adam Doupé. 2021. Everything You Ever Wanted to Know About Bitcoin Mixers (But Were Afraid to Ask). In *FC*. 117–146.

[35] Torben Pryds Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology — CRYPTO '91*. 129–140.

[36] Claudia Pop, Mirela Prata, Marcel Antal, Tudor Cioara, Ionut Anghel, and Ioan Salomie. 2020. An Ethereum-based implementation of English, Dutch and First-price sealed-bid auctions. In *ICCP*. 491–497.

[37] A. W. Roscoe. 1995. CSP and determinism in security modelling. In *S&P*. IEEE Computer Society, 114–127.

[38] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar R. Weippl. 2020. HydRand: Efficient Continuous Distributed Randomness. In *SP*. 73–89.

[39] Gaurav Sharma, Denis Verstraeten, Vishal Saraswat, Jean-Michel Dricot, and Olivier Markowitch. 2021. Anonymous Fair Auction on Blockchain. In *NTMS*. 1–5.

[40] Zeshun Shi, Cees de Laat, Paola Grosso, and Zhiming Zhao. 2023. Integration of Blockchain and Auction Models: A Survey, Some Applications, and Challenges. *IEEE Commun. Surv. Tutorials* 25, 1 (2023), 497–537.

[41] Zeshun Shi, Cees De Laat, Paola Grosso, and Zhiming Zhao. 2023. Integration of Blockchain and Auction Models: A Survey, Some Applications, and Challenges. *IEEE Commun. Surv. Tutorials* 25 (2023), 497–537. Issue 1.

[42] Nirvan Tyagi, Arasu Arun, Cody Freitag, Riad Wahby, Joseph Bonneau, and David Mazières. 2023. Riggs: Decentralized Sealed-Bid Auctions. In *CSS*. 1227–1241.

[43] Shuangke Wu, Yanjiao Chen, Qian Wang, Minghui Li, Cong Wang, and Xiangyang Luo. 2019. CReam: A Smart Contract Enabled Collusion-Resistant e-Auction. *IEEE Transactions on Information Forensics and Security* 14 (2019), 1687–1701.
[44] Jie Xiong and Qi Wang. 2019. Anonymous Auction Protocol Based on Time-Released Encryption Atop Consortium BlockChain. *IJAIT* 09 (2019), 01–16.
[45] Andrew C. Yao. 1982. Protocols for secure computations. In *SFCS*. 160–164.
[46] Zongli Ye, Chin-Ling Chen, Wei Weng, Hongyu Sun, Woei-Jiunn Tsaur, and Yong-Yuan Deng. 2023. An anonymous and fair auction system based on blockchain. *The Journal of Supercomputing* 79 (2023), 13909–13951.
[47] Steve Zdancewic and Andrew C. Myers. 2003. Observational Determinism for Concurrent Program Security. In *CSFW*. IEEE Computer Society, 29.
[48] Yaning Zhang, Youcai Qian, et al. 2022. RANDAO: A DAO working as RNG of Ethereum. https://github.com/randao/randao
[49] Zijian Zhang, Xin Lu, Meng Li, Jincheng An, Yang Yu, Hao Yin, Liehuang Zhu, Yong Liu, Jiamou Liu, and Bakh Khoussainov. 2024. A Blockchain-Based Privacy-Preserving Scheme for Sealed-Bid Auction. *IEEE TDSC* 21 (2024), 4668–4683.
[50] Ke Zhong, Yiping Ma, Yifeng Mao, and Sebastian Angel. 2023. Addax: A fast, private, and accountable ad exchange infrastructure. In *NSDI*. USENIX Association, 825–848.

Acknowledgments

This work was partially supported by the Ethereum Foundation Research Grant FY24-1793. J. Ballweg was supported by a Hong Kong PhD Fellowship (HKPF). Authors are ordered alphabetically.