Double Auction Meets Blockchain: Consensus from Scored Bid-Assignment*

Xiangyu Su[1], Xavier Défago[1], Mario Larangeira[1,2], Kazuyuki Mori[3], Takuya Oda[1,4], Yasumasa Tamura[1,5], and Keisuke Tanaka[1]

[1] Institute of Science Tokyo. 2-12-1 Ookayama Meguro-ku, Tokyo, Japan. `su.x.4029@m.isct.ac.jp`, `defago@comp.isct.ac.jp`, `rebello.m.f72a@m.isct.ac.jp`, `keisuke@comp.isct.ac.jp`.
[2] Input Output, Global. `mario.larangeira@iohk.io`.
[3] Mitsubishi Electric Corporation. 8-1-1 Tsukaguchi-honmachi Amagasaki, Hyogo, Japan. `Mori.Kazuyuki@ab.MitsubishiElectric.co.jp`.
[4] The University of Kitakyushu. 1-1 Hibikino Wakamatsu-ku Kitakyushu, Fukuoka, Japan. `t-oda@kitakyu-u.ac.jp`.
[5] Hokkaido University, Kita 14, Nishi 9, Kita-ku Sapporo, Hokkaido, Japan. `ytamura@ist.hokudai.ac.jp`.

Abstract. A double auction system, where buyers and sellers trade through bids, requires a transparent and immutable mechanism to record allocation results. This demand can be met with robust ledgers that ensure persistence and liveness, as exemplified by the Bitcoin blockchain (EuroCrypt '15). While existing blockchain-aided auction systems often rely on secure smart contracts or layer-2 techniques, this work proposes a more fundamental approach by constructing a provably secure blockchain protocol directly from the computation of bid allocations. The core component is an alternative proof-of-work (PoW) scheme based on a scored generalized multiple assignment problem (SGMAP), integrated into a tailored blockchain protocol. Unlike conventional PoW-based protocols, our leader selection is driven by block scores derived from the SGMAP scoring function, which is designed to be flexible enough to define the difficulty level and accommodate real-life requirements of the underlying double auction system. We prove persistence and a modified liveness property for our design, and present implementation results to validate its robustness and practicality.

Keywords: Double Auctions, Scored Generalized Multiple Assignment Problem, Alternative Proof-of-Work, Blockchain Consensus.

1 Introduction

In a double auction system, multiple buyers and sellers issue (buy or sell) bids to buy or sell some products. Eligible bids are matched, *i.e.*, buy bids are allocated with sell bids, and vice versa, into trading agreements according to some

* This work was supported by the JST CREST under Grant JPMJCR2113, and the JSPS KAKENHI under Grant JP24H00071 and Grant JP21K11882.

predefined conditions, *e.g.*, the buy bid's price is higher than the sell bid's price. A necessary functionality in such a system is to securely record the trading history so users can perform accordingly. In a decentralized environment, a public ledger that ensures consensus is usually used for this purpose, preventing any single party from tampering with the data.

The problem of consensus, in which multiple participants are made to agree on common decisions, has been studied across five decades in the context of distributed systems [22, 23]. Due to the popularity of Bitcoin and other cryptocurrencies, consensus and its embodiment as a blockchain protocol has gained a much broader interest, taking the form of a distributed robust ledger, ensuring consensus through the properties of persistence and liveness [16]. It is natural to consider blockchain as the public ledger in double auction systems. In fact, numerous studies have investigated blockchain-aided double auction systems (with some in the context of peer-to-peer energy trading) [7, 12–14, 19, 24–26, 28, 33, 35, 36]. As we will show in the detailed literature review (Section 1.2), most of these works are built atop secure smart contracts and layer-2 techniques. However, the assumption of smart contracts being secure is unfalsifiable [27], let alone the genuinely higher transaction fee incurred by using smart contacts. In contrast, this work aims to build consensus on blockchain that directly utilizes mechanisms in double auctions, *e.g.*, bidding operations and the allocation of bids. Through this, our protocol can handle more general double auction models (as explained in Section 1.1) and can be proven secure without relying on smart contracts.

1.1 Our Approach and Contributions

The first step is to clarify our double auction model.

Our double auction model. We consider a single-product[1] periodic double auction system, in which the underlying market is cleared periodically so that users can perform product and payment transmission accordingly in real life. Bids are issued to buy/sell the product with three attributes: type (*i.e.*, indicating to buy or to sell), unit-price, and quantity. Each bid is associated with a lifespan attribute, indicating whether it is time-wise available to be matched with other bids. Unlike conventional double auctions, *e.g.*, uniform price auction or k-double auction, we introduce a more flexible mechanism that enables many-to-many assignments between buy and sell bids. This model can find use cases in real-life applications like peer-to-peer energy trading.

Concretely, a buy bid can be matched with multiple sell bids (and vice versa) if the following conditions hold:

1. The price in the buy bid is higher than the price in the sell bid;
2. The sum of the quantity in the multiple sell (buy) bids does not exceed the quantity of the buy (sell) bid;

[1] The multi-product case can be achieved by executing multiple instances of our protocol in parallel.

3. All the bids are within their lifespan.

Based on 2 and 3, we define the *"residual"* (later Definition 5) that inherits the price and lifespan of the original bid while its quantity is the subtraction of the matched parts from the original quantity. Finally, a matched buy-sell bid pair is a trading agreement in which the buyer transfers the payment, and the seller delivers the product. Note that our double auction model can be extended to impose more constraints, *e.g.*, including a preference list for each bid so that it only buys from sellers (sells to buyers) in the list. However, this paper will focus on the three conditions mentioned above, which are the bare minimum requirements for our many-to-many double auction model. Our blockchain protocol will be general enough to cover extensions made to this model.

A brief description of our approach. To integrate double auctions into blockchain protocols, we first define a bid layer to the data structure to support bidding operations (*i.e.*, issuing buy/sell bids). Atop the bid layer, we refine transactions as matched buy-sell bid pairs, and blocks as the containers of bids and transactions, *i.e.*, each block keeps a list of buy/sell matched transactions. The allocation of buy and sell bids (*i.e.*, forming transactions) is formalized into a Scored Generalized Multiple Assignment Problem (SGMAP, which is a many-to-many match design in our auction model). We define a generic scoring function that evaluates assignment results.

Later, by directly associating the SGMAP with our refined blockchain data structure, *i.e.*, directly relating it to our auction model, and lifting the scoring function's domain to the space of blocks, we propose the core component of our blockchain protocol: an alternative proof-of-work (PoW) scheme, namely the proof-of-bid-assignment (PoBA) scheme. Similar to the mempool of transactions in conventional blockchain protocols, we introduce the concept of "bidpool" to cope with double auctions and in accordance with our proposed bid layer. Our PoBA starts with each user maintaining such a pool according to her view of the blockchain. At the beginning of each time slot[2], the user collects bids from the previous slot and updates her bidpool. By sampling the bidpool, the user generates a block with its corresponding score by solving the semanticized SGMAP. We require the user to include the input bid set in her block so that the block and score can be verified by all other users.

Note that we do not require *optimality*, *i.e.*, the best possible combination of buy/sell bids, in block generation. This is due to a unique feature of our protocol: we encourage competition among users so that the protocol will eventually put forth a chain of blocks with the highest overall score (*i.e.*, the highest-score rule). Concretely, for blockchain selection, we first consider a tree structure for each user's intermediate view of block candidates (from multiple users in each time slot). Hence, each branch (from root to leaf) on the tree is a valid blockchain. Then, with an accumulation function defined over the depth of each block on the branch, we extend the score of blocks to the score of branches. Thereby, users

[2] We divide time into discrete units call slots, details can be found in Section 2.

can select the highest-scored branch accordingly, *i.e.*, each branch has a score associated given by the score accumulation of its blocks.

Our security proof is quite involved. We first abstract away from concrete computations in the PoBA by directly sampling blocks with corresponding scores from the distribution determined by the generic scoring function. We model this approach under the universal sampler model [20] (with necessary modifications). Our modeling is similar to how hash computation is modeled (*i.e.*, by queries to a random oracle) in conventional hash-PoW-based blockchain protocols, *e.g.*, Bitcoin [16], and the way that we use the universal sampler is similar to [5]. To justify how suitable this modeling approach is, we provide theoretical insights in Section 5.1 and implement the PoBA scheme under various conditions, *i.e.*, using different initial bid distributions and solving strategies, in Section 6. Then, we prove *persistence* by analyzing the dynamics of honest users' views (*i.e.*, represented by their local trees of blocks) concerning scores (of blocks and branches) following the given (arbitrary) distribution. Moreover, we show *block-liveness*, a variant of the standard liveness property, yields directly from our protocol design. Note that the original liveness property indicates state changes in a protocol. Hence, the adaptation of liveness is necessary and reasonable because in our refined data structure, a single transaction (*i.e.*, a matched buy-sell bid pair) cannot change the state of the protocol as in UTXO-based blockchain protocols, *e.g.*, Bitcoin [16]. In contrast, our protocol changes its state when a block (containing the sets of bids and transactions) is inserted into all honest users' local trees. Thus, the security of our protocol is based on the persistence and block-liveness properties.

Our contribution. It is threefold: (1) a blockchain data structure that supports bidding operations in double auction systems; (2) an alternative PoW scheme (*i.e.*, the PoBA) based on the problem of assigning buy and sell bids concerning a generic scoring function (*i.e.*, the SGMAP); (3) design a blockchain protocol for double auction systems based on the PoBA scheme and prove security for our novel consensus mechanism (*i.e.*, the score-based blockchain selection). Additionally, we implement the PoBA scheme and provide a full network simulation for our protocol. The link can be found in Section 6.

1.2 Related Works

Our literature review focuses on two aspects: (1) blockchain-aided double auction systems; (2) alternative PoW-based blockchain protocols.

Firstly, most works rely on blockchains enabled by smart contracts (SC) or layer-2 techniques [7,12,14,19,25,26,28,33,35,36], with some assuming a trusted execution environment (TEE) [13,24]. While most works focused on implementation and engineering, [28], built atop state channels, provided a (UC-based) security proof that requires stronger liveness than relying only on smart contracts. Our work is unique in that it focuses on the consensus layer of a blockchain-aided auction system, utilizing an alternative PoW scheme (our PoBA scheme) that integrates auction mechanisms, *i.e.*, the computation of bid allocations. We further propose a novel blockchain selection mechanism tailored to PoBA to achieve consensus. Our design is securely proven for key consensus properties.

Table 1: Comparison with related works.

Works	Techniques	Focus of Analysis
[7, 19, 25, 26, 33, 35, 36]	SC & Secure blockchain	SC-based privacy and security
[13, 24]	SC & TEE	Sealed-bids auction & TEE-based security
[12, 14]	SC	Sealed-bids auction & SC programming
[28]	SC & State channel	Dispute resolution via SC (UC-based)
This work	Scored bid-assignment problem & The highest-score rule	Consensus: Persistence & Block-liveness

Secondly, we list several works that inspired our design. The aim here is not to make a direct comparison, but rather to trace the relevant research trends. In [16], the authors provided the first rigorous security proof for Hash-PoW-based blockchain protocols. Weighted variants of such protocols, that can optimize network throughput, were later proven secure in [17, 21]. However, the computationally wasteful mining mechanism in the Hash-PoW highlighted the need for alternative tasks (*i.e.*, alternative PoW schemes) to secure blockchain protocols. Hence, secure protocols based on "useful work" were proposed [4, 11] to leverage computing power for meaningful purposes. While our PoBA design is unique compared to existing works, it incorporates the concept of weight-based consensus from [21] into the generic scoring function and uses the allocation of bids as the useful work. While our consensus mechanism differs from the weight-based ones, we show in Appendix F that our PoBA can also be applied to [21]'s framework by lifting the score distribution via transformation functions [32].

1.3 Organization

The remainder is organized as follows. Section 2 defines the notations and the general model of protocol execution. The following sections present our main contributions: Section 3 introduces the tailored data structure and basic abstractions from our double auction model; Section 4 proposes our blockchain protocol by specifying an SGMAP-based PoBA scheme and the score-based blockchain selection mechanism; Section 5 models the PoBA scheme using a universal sampler and proves the security of our protocol; and finally, Section 6 implements the PoBA scheme under various conditions and validates our analysis approach.

2 Preliminaries

We use λ for the security parameter. For any integer $a \leq b$, let $[a..b] \overset{\Delta}{=} \{a, a+1, \ldots, b\}$. If $a \geq 0$, let $[a]_0 \overset{\Delta}{=} \{0, \ldots, a\}$, and if $a \geq 1$, let $[a] \overset{\Delta}{=} \{1, \ldots, a\}$. Given a set X, $x \overset{\$}{\leftarrow} X$ denotes that x is uniformly and randomly sampled from X, and $x \overset{\mathcal{D}}{\leftarrow} X$ denotes that x is randomly sampled from X following distribution $\mathcal{D}$. Given an algorithm Alg, $x \leftarrow \mathsf{Alg}$ denotes that x is assigned the output of the algorithm Alg on fresh randomness. The $(\mathsf{key}{:}\mathsf{value})$ term denotes a mapping from a key key to the value value. A collision-free hash function is denoted by $H : \{0,1\}^* \to \{0,1\}^\lambda$.

We further employ a digital signature scheme $\mathsf{SIG} \overset{\Delta}{=} (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ that satisfies correctness and existential unforgeability under adaptive chosen message attack. The formal definition is given in Appendix A.1. In particular, we say a signature is valid if $\mathsf{SIG.Verify}$ outputs 1.

Protocol execution model. Based on the formalization methodology and model outlined in [16] (including its subsequent refinements, *e.g.*, [9, 30], that operate under weaker assumptions closer to real-life environments), we specify general settings of our protocol as follows. The protocol adopts the standard Interactive Turing Machines (ITM) Model for its execution [6]. A protocol refers to algorithms for a set of nodes (users) to interact with each other. All corrupted nodes are considered to be controlled by an adversary $\mathcal{A}$ who can read inputs and set outputs for these nodes.

Time and slots. Time is divided into discrete units called slots, indexed by an integer $t \geq 0$. Following [9], users are equipped with (approximately) synchronized clocks that indicate the current slot. The length of each slots is adjusted to be long enough so that any discrepancies between users' local time are insignificant. Here, we assume the existence of a (not necessarily fully) synchronized clock $\mathcal{T}$, secured by the signature scheme SIG with a key pair $(\mathsf{sk}_\mathcal{T}, \mathsf{pk}_\mathcal{T})$. It replies users' queries with $(\sigma_\mathcal{T}, t) \leftarrow \mathcal{T}(m)$ *s.t.* $(\sigma_\mathcal{T}, t) \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathcal{T}, m, t)$, where m is the query message and t is the current slot index. This assumption can be instantiated using the *trusted* checkpoints approach proposed in [8], which was later extended to the *decentralized* setting in [1].

Synchrony. We adapt the δ-synchronous setting [6] to our slot-based execution model, where $\delta \in \mathbb{N}$ represents a known upper bound of the actual network delay, shared by all users. Suppose an honest user sends a message in slot t, the message is guaranteed to be received by all honest users in any slot $t' \geq t + \delta$. We assume the delayed diffuse functionality [9] (described in Appendix A.2).

Rushing adversary. The rushing network adversary can: (1) receive honest user messages first; (2) decide for each recipient whether to inject additional messages; (3) decide the order of message delivery; (4) diffuse its (the adversary's) messages after seeing all honest messages. Further, we need a constraint on the adversary (formally in Assumption 1) to argue the security of our protocol.

Permissionless setting. We adopt the permissionless model from [31]. Specifically, in each slot t, exactly $n_t \in \mathbb{N}$ users participate in the protocol, with a corruption fraction f_t (*i.e.*, $f_t \cdot n_t$ corrupted nodes). Upon joining, an honest user is informed by the protocol of the parameters (n_t, f_t, δ). We assume an honest majority, meaning the corruption fraction satisfies $f_t < 1/2$ for all $t \geq 0$.

Corruption model. This paper focuses on the static corruption model, where honest users cannot be corrupted after spawned. However, as discussed in [18, 30], the static corruption can be relaxed to a constrained adaptive one (*i.e.*, a model permitting corruption only at the end of a slot) by employing key-evolving cryptographic primitives[3]. While [30] extended this approach to support a *fully* adaptive corruption model (*i.e.*, honest users can be corrupted at any point during protocol execution), further research is required to adapt their analysis to our specific protocol.

3 Our Formalization of Blockchain-Aided Auctions

This section starts our formalization by presenting the refined blockchain data structure and abstractions of our double auction model.

3.1 Refined Data Structure

We add a bid layer to support bidding operations, *i.e.*, the issuance of buy/sell bids. Recall our double auction model (in Section 1.1), a bid is issued at a generation slot and will expire at an expiration slot to either: (1) buy a quantity of the product at an initial unit-price (anything lower is acceptable); or (2) sell a quantity of the product at an initial unit-price (anything higher is acceptable). Hence, we define each bid to consist of four attributes: type (*i.e.*, buy or sell), unit-price, quantity, and lifespan (*i.e.*, represented by the generation and expiration slot indices). Additionally, we secure the integrity of bids with a hash function $H(\cdot)$ and a signature scheme $\mathsf{SIG}{=}(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ (from Section 2). Next, a trading agreement, usually called a transaction in the context of blockchain protocols, is redefined as a buy-sell bid pair with the corresponding matched unit-price and matched quantity. We further refine the block's definition to be the container of the sets of bids and transactions with necessary metadata, *e.g.*, the hash of the previous block and signatures.

The definitions. Formally, we define them as follows.

Definition 1 (Bid). *A bid, issued by a user* u *with a key pair* $(\mathsf{sk_u}, \mathsf{pk_u})$ *from the signature scheme* SIG*, is defined as* $\mathsf{bid} \overset{\Delta}{=} (\mathsf{bid_{raw}}, \mathsf{slotInfo}, \mathsf{aux_u})$*, and is associated with an identifier* $\mathsf{bidID}{=}H(\mathsf{bid})$*. The space of bids is denoted by* $\mathbb{BID}$*.*

[3] Proposed in [18] and explored in [30] under the term "erasure model", this approach allows honest parties to securely erase their secret internal state upon corruption.

- $\mathsf{bid_{raw}} \triangleq (\mathsf{type}, p, q)$ *is the bid's raw content where:* $\mathsf{type} \in \{\mathsf{buy}, \mathsf{sell}\}$ *indicates the bid's type (i.e., being a buy/sell bid); and* $p, q > 0$ *denote the bid's unit-price and quantity, respectively;*
- $\mathsf{slotInfo} \triangleq (t_{\mathsf{Gen}}, t_{\mathsf{Exp}}, \mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}})$ *is the bid's lifespan information where:* $t_{\mathsf{Gen}}, t_{\mathsf{Exp}}$ *denotes the bid's generation and expiration slot indices s.t.* $-1 \le t_{\mathsf{Gen}} < t_{\mathsf{Exp}}$[4]*; and* $(\mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}})$ *is the time server's public key and its signature on the bid's raw content and slot indices:* $\sigma_{\mathcal{T}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{\mathcal{T}}, (\mathsf{bid_{raw}}, t_{\mathsf{Gen}}, t_{\mathsf{Exp}}))$*;*
- $\mathsf{aux_u} \triangleq (\mathsf{pk_u}, \sigma_{\mathsf{u}})$ *is the user's public key and her signature on the bid's raw content and lifespan information:* $\sigma_{\mathsf{u}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_u}, (\mathsf{bid_{raw}}, \mathsf{slotInfo}))$.

Definition 2 (Transaction). *A transaction is defined as* $\mathsf{tx} \triangleq (\mathsf{bidID}_1, \mathsf{bidID}_2, p_{\mathsf{tx}}, q_{\mathsf{tx}})$ *where:* $\mathsf{bidID}_1, \mathsf{bidID}_2$ *are the identifiers of two bids of different types (i.e., a buy bid and a sell bid); and* $p_{\mathsf{tx}}, q_{\mathsf{tx}} > 0$ *denote the matched unit-price and quantity. The space of transactions is denoted by* $\mathbb{TX}$.

Definition 3 (Block). *A block, generated by* u *with* $(\mathsf{sk_u}, \mathsf{pk_u})$ *from* SIG*, is defined as* $\mathsf{bk} \triangleq (\mathsf{bk_{raw}}, \mathsf{slotInfo}, \mathsf{aux_u})$ *and is associated with an identifier* $\mathsf{bkID} = H(\mathsf{bk})$*. The space of blocks is denoted by* $\mathbb{BK}$.

- $\mathsf{bk_{raw}} \triangleq (\mathsf{prevHash}, \mathsf{BIDs}, \mathsf{TXs})$ *is the block's raw content where:* $\mathsf{prevHash}$ *is the identifier (i.e., hash) of the block's direct predecessor;* BIDs *denotes a set of bids where each element is in the mapping form* $(\mathsf{bidID}{:}\mathsf{bid}) \in \mathsf{BIDs}$*; and* TXs *denotes a set of transactions where* $\mathsf{tx} \in \mathsf{TXs}$*;*
- $\mathsf{slotInfo} \triangleq (t_{\mathsf{Gen}}, \mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}})$ *is the block's generation information where:* $t_{\mathsf{Gen}} \ge 0$ *denotes the block's generation slot index; and* $(\mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}})$ *is the time server's public key and its signature on the block's raw content and slot index:* $\sigma_{\mathcal{T}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{\mathcal{T}}, (\mathsf{bk_{raw}}, t_{\mathsf{Gen}}))$*;*
- $\mathsf{aux_u} \triangleq (\mathsf{pk_u}, \sigma_{\mathsf{u}})$ *is the user's public key and her signature on the block's raw content and generation information:* $\sigma_{\mathsf{u}} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk_u}, (\mathsf{bk_{raw}}, \mathsf{slotInfo}))$.

For completeness, we recall the definition of a blockchain: a linked list of blocks, on which the first block is called the genesis block, denoted by bk^G, containing initial users' public keys. Later, we will consider a tree structure (to be explained in Section 4.3) to analyze users' local views. In order to distinguish an arbitrary blockchain from the highest-scored one, we use the term "branch" to describe the former and "chain" for the latter.

Definition 4 (Chain). *Let* $||$ *denote block concatenation. That is, for any* $i \ge 0$*, if* $\mathsf{bk}^{i-1}||\mathsf{bk}^{i}$ *(let* $\mathsf{bk}^{-1} = \mathsf{bk}^G$*), the* $\mathsf{prevHash}$ *entry of* bk^i *equals to* bk^{i-1}*'s identifier* $\mathsf{bkID}^{i-1} = H(\mathsf{bk}^{i-1})$*. Then, a blockchain for* $t \ge 0$ *is defined as the concatenation of blocks:* $\mathsf{chain}^t \triangleq \mathsf{bk}^G||\mathsf{bk}^0||\cdots||\mathsf{bk}^t$ *where* bk^G *is the genesis block. Moreover, for any* $\tau \in [t]_0$*, we use* $\mathsf{chain}^{t\lceil\tau}$ *to denote the blockchain without* τ*-rightmost blocks, i.e.,* $\mathsf{chain}^{t\lceil\tau} \triangleq \mathsf{bk}^G||\mathsf{bk}^0||\cdots||\mathsf{bk}^{t-\tau}$.

[4] In particular, we call a bid initial if $t_{\mathsf{Gen}} = -1$, indicating that the bid was issued before the first block was generated. Moreover, we require $t_{\mathsf{Gen}} < t_{\mathsf{Exp}}$ because the bid allocation has at least one slot delay by our protocol design (see Section 4.2).

Concrete configurations and their rationale. Here, we further specify several configurations concerning our double auction model.

Firstly, the bid's definition is extensible by including more attributes, *e.g.*, the preference list mentioned before (in Section 1.1). We can even integrate a reputation system by letting each bid include a signed (by some trusted authorities) reputation score of its issuer, and then design our generic scoring function (to be explained in the following sections) to take the reputation score into consideration. However, the current Definition 1 is sufficient for our protocol design, and hence, the following of this work will proceed under this definition.

Bids, and hence their identifiers, are designed to be unique, unlike transactions, which do not require uniqueness. That is, there can be multiple transactions with the same $(\mathsf{bidID}_1, \mathsf{bidID}_2, p_{\mathsf{tx}}, q_{\mathsf{tx}})$. However, we consider that, in any block, a $(\mathsf{bidID}_1, \mathsf{bidID}_2, p_{\mathsf{tx}})$ tuple appears only once in transactions. This is because if multiple transactions share the same $(\mathsf{bidID}_1, \mathsf{bidID}_2, p_{\mathsf{tx}})$, their q_{tx} can be aggregated, allowing them to be unified into a single transaction.

Moreover, recall that our double auction model aims to enable many-to-many assignments between buy and sell bids. To facilitate this, we introduce the concept of residuals, which represents the remaining quantity of a bid after it has been partially matched with bids of the opposite type. The formal definition is given in the form of a mapping *w.r.t.* a bid and a transaction involving that bid. For simplicity, we inherently assume that all bids and transactions are "valid": the three conditions outlined in our double auction model hold, and signatures and identifiers are correctly computed.

Definition 5 (Residual Bids). *Let* $(\mathsf{bidID}{:}\mathsf{bid})$ *be the mapping form of a bid where* $\mathsf{bid}=(\mathsf{bid}_{\mathsf{raw}}, \mathsf{slotInfo}, \mathsf{aux}_{\mathsf{u}})$ *and* $\mathsf{bid}_{\mathsf{raw}}=(\mathsf{type}, p, q)$*. Given a transaction* $\mathsf{tx}= (\mathsf{bidID}, \cdot, p_{\mathsf{tx}}, q_{\mathsf{tx}})$ *that involves the given bid, we denote the residual bid of* bidID *w.r.t.* tx *with* $(\mathsf{bidID}{:}\mathsf{rbid})$ *s.t.: if* $q{>}q_{\mathsf{tx}}$*,* $\mathsf{rbid}\overset{\Delta}{=}(\mathsf{rbid}_{\mathsf{raw}}, \mathsf{slotInfo}, \mathsf{aux}_{\mathsf{u}})$ *where* $\mathsf{rbid}_{\mathsf{raw}} =(\mathsf{type}, p, q' = q{-}q_{\mathsf{tx}})$*; otherwise (i.e.,* $q{=}q_{\mathsf{tx}}$*),* $\mathsf{rbid}\overset{\Delta}{=} \perp$*.*

We note that tx in the definition should be valid *w.r.t.* bid, *i.e.*, to have $q{\geq}q_{\mathsf{tx}}$. Moreover, when $q{=}q_{\mathsf{tx}}$, we consider the bid is fully matched, *i.e.*, $\mathsf{rbid} = \perp$.

Given another transaction $\mathsf{tx}'{=}(\mathsf{bidID}, \mathsf{bidID}'', p'_{\mathsf{tx}}, q'_{\mathsf{tx}})$, we can extract the residual of $(\mathsf{bidID}{:}\mathsf{rbid})$ by further subtracting q'_{tx} from q', *i.e.*, the new residual bid is defined as $(\mathsf{bidID}{:}\mathsf{rbid}')$: if $q'{>}q'_{\mathsf{tx}}$, $\mathsf{rbid}'_{\mathsf{raw}}\overset{\Delta}{=}(\mathsf{type}, p, q'{-}q'_{\mathsf{tx}})$; otherwise, $\mathsf{rbid}'\overset{\Delta}{=} \perp$. This process can be repeated recursively until the residual quantity reaches 0 or the bid expires. The recursive nature of residual bids (while preserving their identifiers) allows us to save block space by only storing original ("non-residual") bids in the bid sets, *i.e.*, $\mathsf{bidID} = H(\mathsf{bid})$ for any $(\mathsf{bidID}{:}\mathsf{bid}) \in \mathsf{BIDs}$. Users can ambiguously extract the latest quantity of any bid concerning its identifier with the given blockchain. Note that the equation for identifiers above does not hold for residual bids as their quantities have changed.

Concretely, for a blockchain $\mathsf{chain}^t{=}\mathsf{bk}^G||\mathsf{bk}^0||\ldots||\mathsf{bk}^t$ with $t{\geq}0$, let BIDs^i and TXs^i be the bid set and transaction set in each block bk^i for all $i \in [t]_0$, respectively. Note that each $(\mathsf{bidID}{:}\mathsf{bid})$ should exist only once in a bid set on chain^t by our configuration, we denote the set of transactions involving

bidID with $T^t_{\mathsf{bidID}} \overset{\Delta}{=} \{\mathsf{tx} | \mathsf{tx}{=}(\mathsf{bidID}, \cdot, p_{\mathsf{tx}}, q_{\mathsf{tx}}) {\in} \bigcup_{i\in[t]_0} \mathsf{TXs}^i\}$[5] for each $(\mathsf{bidID}{:}\mathsf{bid}) \in \bigcup_{i\in[t]_0} \mathsf{BIDs}^i$. Then, let q be the quantity of bid, the latest quantity of the residual bid associated with bidID, denoted by q', can be derived by the equation (as before, only meaningful when $q' > 0$)

$$q' = q - \sum_{\mathsf{tx} \in T^t_{\mathsf{bidID}}} q_{\mathsf{tx}}. \tag{1}$$

One concern with this configuration (*i.e.*, storing only original bids in blocks) is that block verification may become less efficient because the latest residual of each bid is not explicitly recorded. However, we argue that users can track residuals locally on a block-by-block basis, for example, by updating T_{bidID} with transactions from $\mathsf{TXs}^{t+1} \in \mathsf{bk}^{t+1}$ that involve bidID.

3.2 Abstraction of Bid Assignment

Now, we abstract the process of allocating bids in our double auction model, then present a scored variant of the generalized multiple assignment problem (SGMAP, which is many-to-many) [29]. Precisely, we consider two sets $B{=}\{b_i \overset{\Delta}{=} (\mathsf{B}, p_i, q_i)\}_{i\in[m]}$ and $S{=}\{s_i \overset{\Delta}{=} (\mathsf{S}, p_j, q_j)\}_{j\in[n]}$ (where $m, n{\geq}1$) as the problem's input. Here, each element (*i.e.*, $b{\in}B$ and $s{\in}S$) consists of a flag that indicates in which set the element is in (*i.e.*, B for set B and S for set S, ensuring the two sets are disjoint); and two value attributes *s.t.* $p, q{>}0$. The corresponding space of the sets are denoted by $\mathbb{B}{=}\{B | \forall p, q{>}0\}$ and $\mathbb{S}{=}\{S | \forall p, q{>}0\}$.

Then, a match between $b_i{\in}B$ and $s_j{\in}S$ concerning value attributes can be represented by their indices as (i, j, p_{ij}, q_{ij}) where p_{ij}, q_{ij} denote the assigned values. By considering the attributes in terms of our double auction model (*i.e.*, B for buying, S for selling, p for price, and q for quantity), we directly yield a requirement: $q_{ij}{\geq}0$, in which $q_{ij}{=}0$ indicates there is *no* match between b_i and s_j. Next, we can impose the first two constraints[6] from the model. Recall that a buy bid can be matched to a sell bid if: (1) the buy bid's price is higher than the sell bid's price[7]; (2) the residual quantity of both bids is no less than 0 after being matched together. We formally write these constraints as follows:

$$q_{ij} \geq 0 \wedge p_{ij} \in [p_j, p_i] \text{ if } q_{ij} \neq 0; \qquad \sum_{j=0}^{n-1} q_{ij} \leq q_i \wedge \sum_{i=0}^{m-1} q_{ij} \leq q_j. \tag{2}$$

Finally, we define the scoring function that evaluates assignment results. Let the set of assignment results for the input sets (B, S) be denoted as $A_{B,S}{=}$

[5] We omit writing "or $\mathsf{tx}{=}(\cdot, \mathsf{bidID}, p_{\mathsf{tx}}, q_{\mathsf{tx}})$" in T^t_{bidID} because the two bid identifiers in a transaction are symmetric, and bidID can only exist in one position.

[6] The third one involves lifespan, which will be discussed later concerning the correctness of our PoBA scheme.

[7] In practice, it is convenient to consider a deterministic pricing strategy as in the k-double auction model, *i.e.*, given $k{\in}[0,1]$, set $p_{ij}{=}k{\cdot}p_j{+}(1{-}k){\cdot}p_i$.

$\{(i, j, p_{ij}, q_{ij}) | \forall (i,j) \in [m] \times [n] \wedge (p_{ij}, q_{ij})$ satisfies conditions in Eq. 2$\}$. Then, the generic scoring function is defined over the space of assignment results as $s : \mathbb{A} \to \mathbb{R}$ where $\mathbb{A} = \{A_{B,S} | \forall (B, S) \in \mathbb{B} \times \mathbb{S}\}$.

The next formal definition yields directly from the early explanation.

Definition 6 (The SGMAP). *Given two sets $B=\{(\mathsf{B}, p_i, q_i)\}_{i\in[m]}$ and $S=\{(\mathsf{S}, p_j, q_j)\}_{j\in[n]}$ with $m, n \geq 1$ and any $p, q > 0$. The SGMAP is to find a set $A_{B,S}=\{(i, j, p_{ij}, q_{ij})\}$ that satisfies conditions in Eq. 2 and evaluate the set with the generic scoring function given above, i.e., $s(A_{B,S})$.*

Note that it is easy to find a valid solution to the SGMAP as any random assignment set that satisfies conditions in Eq. 2 is sufficient. In contrast, the difficulty of finding an optimal solution depends on the scoring function. Proven in [29], the optimization problem is NP-complete even when the scoring function is just a linear combination of q_{ij} for $(i, j) \in [m] \times [n]$. We will review this in Section 5.1 and present experiments with concrete examples in Section 6.

4 Our Protocol

This section first semanticizes the SGMAP with our refined blockchain data structure to yield an alternative PoW scheme, namely proof-of-bid-assignment (PoBA). Then, by specifying the process of users preparing their input bid set (to the PoBA) and the selection of blockchain (after generating blocks from the PoBA), we build a blockchain protocol based on the PoBA scheme.

4.1 The PoBA Scheme

We first semanticize the definition of SGMAP (Definition 6) with the raw content of bids (Definition 1) and transactions (Definition 2).

Definition 7 (The SGMAP, Semanticized). *Let $\mathsf{Bs} = \{(\mathsf{bidID}{:}\mathsf{b}) | \mathsf{b} \in \{\mathsf{bid}, \mathsf{rbid}\}\}$ be a set of (residual) bids where bidID is the identifier of the original bid corresponding to b. Extract the buy and sell bid subset:*

$$B=\{(\mathsf{bidID}_i{:}\mathsf{b}_i) | \mathsf{b}_{\mathsf{raw},i} = (\mathsf{buy}, p_i, q_i)\}_{i\in[m]},$$
$$S=\{(\mathsf{bidID}_j{:}\mathsf{b}_j) | \mathsf{b}_{\mathsf{raw},j} = (\mathsf{sell}, p_j, q_j)\}_{j\in[n]},$$

s.t. $\mathsf{Bs} = B \cup S$ and $B \cap S = \emptyset$. The SGMAP is to find a transaction set $\mathsf{TXs}=\{(\mathsf{bidID}_i, \mathsf{bidID}_j, p_{ij}, q_{ij})\}$ so that p_{ij}, q_{ij} satisfies conditions in Eq. 2. By abuse of notations, the score of the transaction set is $s(\mathsf{TXs})$ (i.e., $s : \mathbb{TX}^* \to \mathbb{R}$).

Next, we want users to generate blocks by solving the semanticized SGMAP in our PoBA. Hereby, we lift the range of the generic scoring function from transaction sets to blocks by considering a transforming function $T{:}\mathbb{R} \times \{0,1\}^* \to \mathbb{R}$. Concretely, the score of a given block that embeds a transaction set, *i.e.*, $\mathsf{TXs} \in \mathsf{bk}$, is denoted by $s_{bk}(\mathsf{bk}) = T(s(\mathsf{TXs}), \mathsf{bk} \backslash \mathsf{TXs})$ (*i.e.*, $s_{bk}{:}\mathbb{BK} \to \mathbb{R}$) where $\mathsf{bk} \backslash \mathsf{TXs} \overset{\Delta}{=} (\mathsf{prevHash}, \mathsf{BIDs}, \mathsf{slotInfo}, \mathsf{aux_u})$ is the other content in the block (by Definition 3).

Finally, the PoBA scheme $\mathsf{PoBA} \overset{\Delta}{=} (\mathsf{Solve}, \mathsf{Verify})$ is specified as follows.

Construction 1 (The PoBA)**.** *Let* $H{:}\{0,1\}^*{\to}\{0,1\}^\lambda$ *be a collision-free hash function, and let* $s_{bk}{:}\mathbb{BK}{\to}\mathbb{R}$ *be a generic scoring function that evaluates blocks. For any user* u*, denote her blockchain at the beginning of slot* $t{+}1{\geq}0$ *with* chain^t $=\mathsf{bk}^G||\cdots||\mathsf{bk}^t$ *(*$\mathsf{chain}^{-1}\overset{\Delta}{=}\mathsf{bk}^G$*). She performs* Solve *and* Verify *s.t.:*

– $\mathsf{Solve}(\mathsf{chain}^t, \mathsf{Bs})$ *takes as input the blockchain and a set of bids*[8]*. It outputs an extended blockchain* $\mathsf{chain}^t||\mathsf{bk_u}$ *where* $\mathsf{bk_u}{=}((H(\mathsf{bk}^t), \mathsf{BIDs}, \mathsf{TXs}), (t{+}1, \mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}}), (\mathsf{pk_u}, \sigma_\mathsf{u}))$ *s.t.:*
  - *For any* $(\mathsf{bidID{:}b}){\in}\mathsf{Bs}$*, if* $H(\mathsf{b}){=}\mathsf{bidID} \wedge (\mathsf{bidID{:}b}){\notin}\bigcup_{i\in[t]_0}\mathsf{BIDs}^i$*, the bid set* $\mathsf{BIDs} \in \mathsf{bk_u}$ *contains* $(\mathsf{bidID{:}b})$*;*
  - TXs *is a valid solution to the semanticized SGMAP regarding* Bs*;*
  - *Signatures* $(\sigma_{\mathcal{T}}, \sigma_\mathsf{u})$ *are generated according to Definition 3.*

– $\mathsf{Verify}(\mathsf{chain}'^{t+1})$ *takes as input a blockchain. Parse* $\mathsf{chain}'^{t+1}{=}\mathsf{chain}'^t||\mathsf{bk_{u'}}$[9]*, i.e.,* $\mathsf{bk_{u'}}$ *is generated by* u'*. Parse* $\mathsf{bk_{u'}}{=}((\mathsf{prevHash}, \mathsf{BIDs}, \mathsf{TXs}), (t_\mathsf{Gen}, \mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}}), (\mathsf{pk_{u'}}, \sigma_\mathsf{u'}))$*. It outputs* $(1, s_{bk}(\mathsf{bk_{u'}}))$ *if the following conditions hold (i.e., the input blockchain is valid); Otherwise, it outputs* $(0, \perp)$*.*
  1. *The blockchain being extended is valid:* $(1,\cdot){\leftarrow}\mathsf{Verify}(\mathsf{chain}'^t)$*. In particular, we extend this verification to the genesis block, i.e.,* $\mathsf{Verify}(\mathsf{chain}^{-1}) = 1$*;*
  2. *Parse* $\mathsf{chain}'^t{=}\mathsf{bk}^G||\cdots||\mathsf{bk}'^t$*, we have* $\mathsf{prevHash}{=}H(\mathsf{bk}'^t)$*;*
  3. *The bid set satisfies, for each* $(\mathsf{bidID{:}bid}){\in}\mathsf{BIDs}$*, that: (1)* $H(\mathsf{bid}) = \mathsf{bidID}$*; (2)* $(\mathsf{bidID{:}bid}){\notin}\bigcup_{i\in[t]_0}\mathsf{BIDs}'^i$*; (3)* bid*'s lifespan* $(t_\mathsf{Gen}, t_\mathsf{Exp})$ *satisfies* $t_\mathsf{Gen}{\leq}t$ $<t_\mathsf{Exp}$[10]*; (4) the signatures* $(\sigma_{\mathcal{T}}, \sigma_{\mathsf{u''},\mathsf{bid}})$ *in* bid *are valid;*
  4. *The transaction set satisfies, for each* $(\mathsf{bidID}_1, \mathsf{bidID}_2, p_\mathsf{tx}, q_\mathsf{tx}) \in \mathsf{TXs}$*, that: (1)* $\mathsf{bidID}_1, \mathsf{bidID}_2 \in \bigcup_{i\in[t]_0}\mathsf{BIDs}'^i \cup \mathsf{BIDs}$ *where* $\mathsf{BIDs}'^i \in \mathsf{bk}'^i \in \mathsf{chain}'^t$*; (2) Extract all transactions from* $\mathsf{chain}'^t||\mathsf{bk_{u'}}$ *for* $\mathsf{bidID}_1, \mathsf{bidID}_2$*:*

$$T^{t+1}_{\mathsf{bidID}_1}{=}\{\mathsf{tx}|\mathsf{tx}{=}(\mathsf{bidID}_1, \cdot, p_\mathsf{tx}, q_\mathsf{tx}){\in}\bigcup_{i\in[t]_0}\mathsf{TXs}'^i \cup \mathsf{TXs}\},$$
$$T^{t+1}_{\mathsf{bidID}_2}{=}\{\mathsf{tx}|\mathsf{tx}{=}(\cdot, \mathsf{bidID}_2, p_\mathsf{tx}, q_\mathsf{tx}){\in}\bigcup_{i\in[t]_0}\mathsf{TXs}'^i \cup \mathsf{TXs}\}.$$

  The conditions given by Eq. 2 hold for both bidID_j*,* $j \in \{1,2\}$*, i.e.,*

$$p_\mathsf{tx}{\in}[p_2, p_1] \wedge \forall j{\in}\{1,2\}, \sum_{\{\mathsf{tx}\in T^{t+1}_{\mathsf{bidID}_j}\}} q_\mathsf{tx} \leq q_j,$$

  where (p_j, q_j) *is the price and quantity of* $(\mathsf{bidID}_j{:}\mathsf{bid}_j){\in}\bigcup_{i\in[t]_0}\mathsf{BIDs}'^i \cup \mathsf{BIDs}$*. Note that* bid_1 *is for buying and* bid_2 *is for selling, hence,* $p_2{\leq}p_1$*.*
  5. *The generation slot* $\mathsf{bk_{u'}}.t_\mathsf{Gen}{=}t{+}1$*, and the signatures* $(\sigma_{\mathcal{T}}, \sigma_\mathsf{u'})$ *are valid.*

[8] Bs is the user's local input to solve the semanticized SGMAP, which may contain residual bids given by chain^t. Intuitively, the bid set BIDs in the newly generated block should contain non-residual bids from Bs that are not included in chain^t.

[9] u' and u may not share the same view of the blockchain, *i.e.*, $\mathsf{chain}'^t{\neq}\mathsf{chain}^t$.

[10] We require valid blocks of slot $t{+}1$ only contain bids issued up to the end of slot t to prevent on-the-fly bid issuance for artificially boosting block scores.

We say the PoBA is correct, if $\mathsf{Verify}(\mathsf{chain}^t||\mathsf{bk}_\mathsf{u}){=}(1, s_{bk}(\mathsf{bk}_\mathsf{u}))$ for any *valid* Bs and $(\mathsf{chain}^t||\mathsf{bk}_\mathsf{u}){\leftarrow}\mathsf{Solve}(\mathsf{chain}^t, \mathsf{Bs})$. The preparation of the input bid set Bs and its validity will be clarified below. On a high level, and consistent with our configurations (Section 3.1), each user can extract residual bids locally and maintain a pool of bids (referred to as the "bidpool" Pool) based on their view of the blockchain. However, to address practical constraints, such as limited block sizes, we introduce an inherent parameter $\mathsf{N}{\in}\mathbb{N}$ that imposes an upper bound on the size of $\mathsf{Bs} \subseteq \mathsf{Pool}$, *i.e.*, $|\mathsf{Bs}| \leq \mathsf{N}$.

4.2 Bidpool Update and Validation

A bidpool is analogous to the mempool in conventional blockchain protocols, with the difference that a bidpool stores bids instead of transactions. A user's bidpool is maintained by two aspects: (1) the user's view of the blockchain, *i.e.*, extracting the latest residual bids and removing them from the bidpool if their residual quantity reaches 0; (2) the non-residual bids issued in previous slots but not included in the user's bidpool.

Concretely, for each user, at the beginning of slot $t{+}1{\geq}0$, denote her bidpool and her blockchain of slot t with Pool^t and chain^t ($\mathsf{Pool}^{-1}{=}\emptyset$ and $\mathsf{chain}^{-1}{=}\mathsf{bk}^G$), respectively. Moreover, we write the set that consists of bids in the second aspect.

$$B^t \overset{\Delta}{=} \{(\mathsf{bidID}{:}\mathsf{bid}) | H(\mathsf{bid}){=}\mathsf{bidID} \wedge \mathsf{bid}.t_\mathsf{Gen}{\leq}t{<}\mathsf{bid}.t_\mathsf{Exp} \wedge (\mathsf{bidID}{:}\mathsf{bid}){\notin}\mathsf{Pool}^t\} \quad (3)$$

Here, following the requirement in block verification (Verify in Construction 1), the set B^t only collects bids with $t_\mathsf{Gen}{\leq}t$. For $t{=}{-1}$, we denote $B^{-1}\overset{\Delta}{=}\{(\mathsf{bidID}{:}\mathsf{bid})|$ $\mathsf{bid}.t_\mathsf{Gen}{=}-1\}$, *i.e.*, consisting of initial bids (Definition 1).

Then, the update algorithm $\mathsf{UpdatePool}(\mathsf{Pool}^t, \mathsf{chain}^t, B)$ outputs the bidpool of slot $t{+}1$, *i.e.*, Pool^{t+1}. We specify $\mathsf{UpdatePool}$ in Algorithm 1. Finally, we say Pool^{t+1} is valid *w.r.t.* chain^t if there are no duplicates or conflicts, as formally defined in Definition 8. Therefore, given chain^t, any subset $\mathsf{Bs}^{t+1} \subseteq \mathsf{Pool}^{t+1}$ is considered a valid input set for the PoBA scheme in slot $t{+}1$.

Definition 8 (Validity of Bidpools). *Let* Pool^{t+1} *be a user's bidpool in slot* $t{+}1{\geq}0$*, and let* $\mathsf{chain}^t{=}\mathsf{bk}^G||\ldots||\mathsf{bk}^t$ *be the valid blockchain in her view at the beginning of slot* $t{+}1$ *(by Construction 1).* Pool^{t+1} *is valid w.r.t.* chain^t *if the following conditions hold for any* $(\mathsf{bidID}{:}\mathsf{b}){\in}\mathsf{Pool}^{t+1}$*:*

- $\mathsf{b}.p, \mathsf{b}.q{>}0$*, and* $\mathsf{b}.t_\mathsf{Gen}{\leq}t{<}\mathsf{b}.t_\mathsf{Exp}$*;*
- *If* $\exists(\mathsf{bidID}{:}\mathsf{bid}) \in \mathsf{chain}^t$*,* b *is identical to* bid *except with* $\mathsf{b}.q$ *is computed by Eq. 1 concerning* $\mathsf{bid}.q$[11]*;*
- *Otherwise, i.e.,* b *is non-residual,* $H(\mathsf{b}){=}\mathsf{bidID}$ *and* $(\mathsf{b}.\sigma_\mathcal{T}, \mathsf{b}.\sigma_\mathsf{u})$ *are valid.*

It is straightforward to see that, given any valid Pool^t (*w.r.t.* any chain^{t-1}), a valid chain^t, and B^t as defined by Eq. 3, the Pool^{t+1} obtained from $\mathsf{UpdatePool}$ is valid *w.r.t.* chain^t. This holds even when chain^t deviates from chain^{t-1} (*i.e.*, $\mathsf{chain}^{t\lceil 1}{\neq}\mathsf{chain}^{t-1}$) due to unsettled blockchain selection.

[11] $\mathsf{b}.q{=}\mathsf{bid}.q$ when the transaction set involving bidID (*i.e.*, T^t_bidID) on chain^t is empty.

Algorithm 1: In any $t+1 \geq 0$, $\mathsf{UpdatePool}$ is parameterized by a bidpool Pool^t, a blockchain chain^t, and a set of bids B^t given by Eq. 3.

```
   // Parse $\mathsf{chain}^t = \mathsf{bk}^G || \mathsf{bk}^0 || \ldots || \mathsf{bk}^t$.
   // For any $i \in [t]_0$, recall $\mathsf{bk}^i_{\mathsf{raw}} = (H(\mathsf{bk}^{i-1}), \mathsf{BIDs}^i, \mathsf{TXs}^i)$.
1  function $\mathsf{UpdatePool}(\mathsf{Pool}^t, \mathsf{chain}^t, B^t)$;
2  if $t = -1$ then
3      Return $\mathsf{Pool}^0 = B^{-1}$ ;
       // $\mathsf{Pool}^{-1}$ is empty, and no bids in $\mathsf{chain}^{-1}$.
4  else
5      Set $\mathsf{Pool}^{t+1} = \mathsf{Pool}^t \cup B^t$ ;
       // $\mathsf{Pool}^t \cap B^t = \emptyset$ by definition.
6      for $(\mathsf{bidID}:\mathsf{bid}) \in \bigcup_{i \in [t]_0} \mathsf{BIDs}^i$ do
7          if $(\mathsf{bidID}:\mathsf{bid}) \notin \mathsf{Pool}^{t+1}$ then
8              Add $(\mathsf{bidID}:\mathsf{bid})$ to $\mathsf{Pool}^{t+1}$ ;
               // $\mathsf{chain}^t$ may contain previously unnoticed bids.
9          end
10     end
       // Note that $\mathsf{Pool}^{t+1}$ inherits residual bids from $\mathsf{Pool}^t$.
       // To distinguish, denote with $\mathsf{b} \in \{\mathsf{bid}, \mathsf{rbid}\}$.
11     for $(\mathsf{bidID}:\mathsf{b}) \in \mathsf{Pool}^{t+1}$ do
12         if $\mathsf{b}.t_{\mathsf{Exp}} < t+1$ then
13             Remove $(\mathsf{bidID}:\mathsf{b})$ from $\mathsf{Pool}^{t+1}$ ;
14         end
15         Let $T^t_{\mathsf{bidID}} = \{\mathsf{tx} | \mathsf{tx} = (\mathsf{bidID}, \cdot, p_{\mathsf{tx}}, q_{\mathsf{tx}}) \in \bigcup_{i \in [t]_0} \mathsf{TXs}^i\}$ ;
           // The case of $\mathsf{tx} = (\cdot, \mathsf{bidID}, p_{\mathsf{tx}}, q_{\mathsf{tx}})$ follows identically.
16         Compute $q' = \mathsf{b}.q - \sum_{\mathsf{tx} \in T^t_{\mathsf{bidID}}} q_{\mathsf{tx}}$ ;
17         if $q' > 0$ then
18             Set $\mathsf{b}' = \mathsf{b}$, except with $\mathsf{b}'.q = q'$ ;
19         else
20             Remove $(\mathsf{bidID}:\mathsf{b})$ from $\mathsf{Pool}^t$;
21         end
22     end
23     Return $\mathsf{Pool}^{t+1}$
24 end
```

4.3 Score-Based Blockchain Selection

Alongside correctness, the difficulty level is another crucial property for alternative PoW schemes, *i.e.*, users must contribute enough computing power to generate valid blocks. Otherwise, they may generate massive amounts of blocks in a short time period so that the network fails to finalize on a chain of blocks. In contrast, as in Section 3.2, finding a valid SGMAP solution (hence, a PoBA block) does not require intensive computing power, *i.e.*, no difficulty.

We tackle this difference by proposing a novel blockchain selection mechanism that encourages users to compete using their blocks (*w.r.t.* scores). One potential challenge arises when the overall bidpool size is insufficient, *i.e.*, there are not enough input bids to even reach the desired optimization difficulty of the SGMAP. To mitigate this, we suggest issuing "dummy" bids in a publicly verifiable manner, *e.g.*, using verifiable random functions over outputs from a publicly available randomness beacon. This approach may also facilitate the dynamic adjustment of the PoBA difficulty.

Concretely, since each user may generate and diffuse multiple valid blocks in each slot, we utilize a directed tree (precisely, forest due to potentially missing blocks) to store blocks locally. Our score-based selection requires *honest* users to extend their block-tree with newly received blocks and select the highest-scored branch on the tree as their blockchain, *i.e.*, the highest-score rule. Hereby, we further extend the score of blocks to branches to support this mechanism.

Starting with definitions, we first consider a master-tree at the end of slot $t{\geq}0$, denoted by mtree^t, that contains all valid blocks generated (*not* diffused) by users up to slot t. It is a directed tree with the genesis block bk^G as the root. Its vertices correspond to blocks, and edges correspond to the hash link between blocks. As in graph theory: (1) vertex height is defined as the number of edges from the vertex to the root; (2) tree height is defined as the number of edges in the longest path from a leaf vertex to the root. Hence, mtree^t is of height $t{+}1$, and blocks are generated in the same slot if vertices in mtree^t are of the same height. Because we assume a rushing adversary controlling block diffusion (while constrained by the bound δ as detailed in Section 2), honest users may only see a part of the master-tree. Hence, a user's view, denoted by $\mathsf{tree}_{\mathsf{u}}^t$, is a sub-tree of the master tree.

Definition 9 (Master-Tree and User-Tree). *Let bk^G be the genesis block. For any $\ell{\geq}0$ and all $i{\in}[n]$ where $n{\geq}1$ is the number of users participating the protocol in slot $t{\in}[\ell]_0$, let $BK_i^t{=}\{\mathsf{bk}_i^t\}$ denote the set of valid blocks generated by user u_i. Then, $BK^t{=}\bigcup_{i\in[n]} BK_i^t$ denotes the set of all valid blocks generated in slot t by all users. The master-tree of slot ℓ is defined as $\mathsf{mtree}^\ell{=}(V,E)$ where $V{=}\bigcup_{t\in[-1..\ell]} BK^t$ and $E{=}\bigcup_{t\in[\ell]_0}\{(\mathsf{bk}^{t-1},\mathsf{bk}^t)|\mathsf{bk}^t.\mathsf{prevHash}{=}H(\mathsf{bk}^{t-1})\}$ with $\mathsf{bk}^{-1}{=}\mathsf{bk}^G$ being the only element in BK^{-1}. A user-tree of u, denoted by $\mathsf{tree}_{\mathsf{u}}^\ell{=}(V_{\mathsf{u}},E_{\mathsf{u}})$, satisfies $\mathsf{tree}_{\mathsf{u}}^\ell{\subseteq}\mathsf{mtree}^\ell$.*

Next, we define branches *w.r.t.* a given block-tree (master or user), which is a chain of blocks (Definition 4).

Definition 10 (Branch on Tree). *Let $G^\ell{=}(V^\ell,E^\ell)$ be a block-tree (master or user) of slot $\ell{\geq}0$. A branch is defined as $\mathsf{branch}^t{=}\mathsf{bk}^G||\mathsf{bk}_{i_1}^0||\ldots||\mathsf{bk}_{i_t}^t$ ($t{\leq}\ell$) s.t. $\mathsf{branch}^t{\subseteq}G^\ell$ and the vertex represents $\mathsf{bk}_{i_t}^t$ on G^ℓ is a leaf vertex.*

We later distinguish the notations of branch and chain by using branch^t for an arbitrary branch on a given block-tree, and chain^ℓ for the highest-scored branch. Next, we define the score of branches by introducing an accumulating function $acc{:}\mathbb{N}{\to}\mathbb{R}$ *s.t.* it takes as input the height of vertices on the given branch.

Definition 11 (Score of Branches and Highest-Score Rule). *Let $\mathsf{branch}^t{\subseteq}$ G^ℓ be a branch as in Definition 10 where $\ell{\geq}0$ and $t{\in}[\ell]_0$. Parse it with $\mathsf{branch}^t{=}$*

$\mathsf{bk}^G||\mathsf{bk}^0_{i_1}||\ldots||\mathsf{bk}^t_{i_t}$. *The score of* branch^t *is defined as follows:*

$$S_{\mathsf{branch}^t}=\sum_{j=0}^{t} acc(t)\cdot s_{bk}(\mathsf{bk}^j). \tag{4}$$

Therefore, our highest-score rule requires honest users with $\mathsf{tree}^{\ell}_{\mathsf{u}}$ *to adopt the blockchain that satisfies* $\mathsf{chain}^{\ell}\overset{\Delta}{=}\arg\max_{\mathsf{branch}^t\subseteq\mathsf{tree}^{\ell}_{\mathsf{u}}} S_{\mathsf{branch}^t}$.

It is important to note that block or branch scores may not be strictly ordered for certain scoring functions, *e.g.*, the concrete example we provided for implementation in Section 6. However, it is convenient to consider the hash value of blocks or branches as a tiebreaker, *e.g.*, with the lower one being preferred.

4.4 Protocol Description

The execution starts at time 0 slot index $t{=}0$, and the genesis block bk^G, containing public keys of initial users, is known to all users.

Considering our periodic auction model, each user u start their slot $t{+}1{\geq}0$ following the instruction of the global clock $\mathcal{T}$. Denote the view of u on her bidpool and user-tree at the end of slot t with $\mathsf{Pool}^t_{\mathsf{u}'}$ and $\mathsf{tree}^t_{\mathsf{u}'}$, respectively (where $\mathsf{Pool}^{-1}_{\mathsf{u}'}{=}\emptyset$ and $\mathsf{tree}^{-1}_{\mathsf{u}'}{=}\{\mathsf{bk}^G\}$). Note that due to our permissionless setting, u may not participate in slot t. Hence, we use u' as the subscript to emphasize that u may obtain her bidpool and user-tree from another user. Moreover, we require $\mathsf{PoBA.Verify}(\mathsf{branch}){=}(1,\cdot)$ for any $\mathsf{branch}{\subseteq}\mathsf{tree}^t_{\mathsf{u}'}$, hence, the user can select her blockchain $\mathsf{chain}^t_{\mathsf{u}'}$ following the highest-score rule (Definition 11). Denote u's view on the non-residual bid set with B^t_{u} (by Eq. 3).

The user first updates her bidpool by executing $\mathsf{Pool}^{t+1}_{\mathsf{u}}{\leftarrow}\mathsf{UpdatePool}(\mathsf{Pool}^t_{\mathsf{u}'},$ $\mathsf{chain}^t_{\mathsf{u}'}, B^t_{\mathsf{u}})$. She can then run $\mathsf{PoBA.Solve}$ on several $\mathsf{Bs}{\subseteq}\mathsf{Pool}^{t+1}_{\mathsf{u}}$ instances, with the inherent upper bound parameter *s.t.* $|\mathsf{Bs}|{\leq}\mathsf{N}$, to obtain multiple block candidates. Let $BK^{t+1}_{\mathsf{u}}\overset{\Delta}{=}\{\mathsf{bk}^{t+1}_{\mathsf{u},j}\}_{j\in[\mathsf{ite}]}$ be the set of candidates where $\mathsf{chain}^t_{\mathsf{u}'}||\mathsf{bk}^{t+1}_{\mathsf{u},j}\leftarrow$ $\mathsf{PoBA.Solve}(\mathsf{chain}^t_{\mathsf{u}'}, \mathsf{Bs}_j)$, and $\mathsf{ite}{\in}\mathbb{N}$ is the maximum number of $\mathsf{PoBA.Solve}$ iterations. Note that ite is bounded by the computing power of the user and the length of each slot. If the user is honest, she only diffuses the highest-scored candidate $\mathsf{bk}^{*t+1}_{\mathsf{u}}\overset{\Delta}{=}\arg\max_{\mathsf{bk}^{t+1}_{\mathsf{u},j}\in BK^{t+1}_{\mathsf{u}}} s_{bk}(\mathsf{bk}^{t+1}_{\mathsf{u},j})$ with the corresponding blockchain $\mathsf{chain}^{t+1}_{\mathsf{u}}\overset{\Delta}{=}\mathsf{chain}^{t-1}_{\mathsf{u}'}||\mathsf{bk}^{*t+1}_{\mathsf{u}}$.

Additionally, each user helps propagate other user's blockchains. A significant concern is the potential strain on network bandwidth. However, it is crucial for our security analysis (as in Lemma 1) that users propagate, at a minimum, their locally highest-scored blockchain in each slot. To reduce network traffic, we propose that during each slot, users propagate a received blockchain only if it surpasses their current highest-scored blockchain. Additionally, users should avoid retransmitting identical information to prevent redundancy.

Finally, the update process of tree-view is specified with $\mathsf{UpdateTree}$ (Algorithm 2). At the end of slot $t{+}1$, u is responsible for reporting her confirmed blockchain $\mathsf{chain}_{\mathsf{u}}^{t+1\lceil\tau}\subseteq\mathsf{tree}_{\mathsf{u}}^{t+1}$, to the protocol *w.r.t.* a parameter τ (to be estimated in Section 5.2). The description of our protocol can be found in Appendix B, and discussions concerning the incentive model in Appendix C.

Algorithm 2: In any slot $t{+}1\geq 0$, $\mathsf{UpdateTree}$ is parameterized by a user-tree tree^{t+1} and a blockchain candidate $\mathsf{chain}^{t'}$.

```
1  function $\mathsf{UpdateTree}(\mathsf{tree}^{t+1}, \mathsf{chain}^{t'})$;
2  Parse $\mathsf{tree}^{t+1} = \{V, E\}$;
   // Verify the input blockchain candidate.
3  Run $(b, \cdot) \leftarrow \mathsf{PoBA.Verify}(\mathsf{chain}^{t'})$;
4  if $b = 0$ then
5      Return $\mathsf{tree}^{t+1}$
6  else
       // $\mathsf{chain}^{t'}$ is valid with $t' = t{+}1$.
       // Parse $\mathsf{chain}^{t'}{=}\mathsf{chain}^{t+1}{=}\mathsf{bk}^{G}||\mathsf{bk}^{0}||\ldots||\mathsf{bk}^{t}||\mathsf{bk}^{t+1}$.
7      for $i \in [t]_0$ do
           // Find the first block not in tree.
8          if $\mathsf{bk}^{i-1} \in \mathsf{tree}^{t+1}$ and $\mathsf{bk}^{i} \notin \mathsf{tree}^{t+1}$ then
9              Set $V' = V \cup \{\mathsf{bk}^{j}\}_{j\in[i..t+1]}$;
10             Set $E' = E \cup \{(\mathsf{bk}^{j}, \mathsf{bk}^{j+1})\}_{j\in[i-1..t]}$;
11             Return $\mathsf{tree}^{t+1} = (V', E')$
12         end
13     end
14 end
```

5 Security Analysis

The first step of our analysis is to model the computation in the PoBA scheme. We follow the approach similar to [5] under the universal sampler model [20].

5.1 Modeling the PoBA

Recall that PoW can be modeled by queries to the random oracle [16]. Extending this methodology, we model the computation of the PoBA as queries to a modified universal sampler [5, 20], which is an "advanced" random oracle that can sample from any given distribution. This approach abstracts away from concrete algorithms and focus on the distribution of block scores instead. This modeling choice is justified by the following observations: (1) for an appropriately chosen scoring function, *e.g.*, the linear combination of assigned quantities, the optimization version of the SGMAP is proven to be NP-complete [29]; (2) users need to sample an N-size bid set as the input to the PoBA from their bidpools in which stochasticity arises due to the uncertainty in problem inputs [10]. We will validate this approach with a concrete scoring function in Section 6.

Now, we show the definition of the modified universal sampler [5] and detail the interaction between users and the universal sampler .

Definition 12 (The Modified Universal Sampler [5]). *A universal sampler consists of algorithms* $\mathsf{US} \overset{\Delta}{=} (\mathsf{Setup}, \mathsf{Sample})$ *that are performed as follows.*

- $\mathsf{Setup}(1^\lambda)$ *takes the security parameter* λ *and outputs sampler parameters* U*;*
- $\mathsf{Sample}(U, \mathsf{d}, \beta)$ *takes the parameters* U*, the program description* d *with a random seed for the program to generate samples. It outputs induced samples* p_{d}*.*

We specify the program description above with $\mathsf{d} \overset{\Delta}{=} (\mathsf{PoBA.Solve}, s_{bk})$. In any slot $t{+}1 \geq 0$, a user u queries the universal sampler with $\beta \overset{\Delta}{=} (\mathsf{Pool}_{\mathsf{u}}^{t+1}, \mathsf{chain}_{\mathsf{u}}^{t}; r_{\mathsf{u}}^{t+1})$ (*i.e.*, we write the randomness explicitly). Following the conventional modeling of computational power [16,21], we assume that, in each slot, each user can make up to $q{>}0$ queries to the universal sampler. This assumption derives from the limited length of slots in the *protocol*, where each user can perform $\mathsf{ite} \leq q$ executions of $\mathsf{PoBA.Solve}$ per slot. The total number of queries in a slot is bounded above by $Q \in \mathbb{N}$, which includes less than f fraction from the adversary, *i.e.*, $Q_{\mathcal{A}} < f \cdot Q$. We further clarify that the network or the adversary cannot delay queries, given that it is local oracle access.

Then, we require the universal sampler to have the single property of randomly sampling a block candidate bk *s.t.* its score $s_{bk}(\mathsf{bk})$ follows the distribution $\mathcal{D}$ determined by the scoring function $s_{bk}(\cdot)$. Denote the score space with $\mathbb{S} \overset{\Delta}{=} s_{bk}(\mathbb{BK})$, we have $s_{bk}(\mathsf{bk}) \overset{\mathcal{D}}{\leftarrow} \mathbb{S}$. Moreover, the probability density function and the distribution function of $\mathcal{D}$ are denoted by $f(\cdot)$ and $F(\cdot)$ that satisfy $F(x) = \Pr[X \leq x] = \int_{\mathsf{smin}}^{x} f(t)dt$.

A Universal Sampler that Models $\mathsf{PoBA.Solve}$

In any slot $t{+}1 \geq 0$, run $U \leftarrow \mathsf{US.Setup}(1^\lambda)$. Let $\mathbb{L}_{\mathsf{u}}^{t+1} = \{(\cdot,\cdot,\cdot,\cdot,\cdot)\}$ be the list kept by the universal sampler for user u, and let U^{t+1} denote the set of all users. We define $U \overset{\Delta}{=} \bigcup_{\mathsf{u} \in \mathsf{U}^{t+1}} \mathbb{L}_{\mathsf{u}}^{t+1}$. The total size of lists is upper bounded by $Q \in \mathbb{N}$: $|U| \leq Q$.
On a query $(\mathsf{Pool}_{\mathsf{u}}^{t+1}, \mathsf{chain}_{\mathsf{u}}^{t}, r_{\mathsf{u}}^{t+1})$ **from** u**:**

- If: there exists a tuple $(\mathsf{Pool}_{\mathsf{u}}^{t+1}, \mathsf{chain}_{\mathsf{u}}^{t}, r_{\mathsf{u}}^{t+1}, \mathsf{bk}_{\mathsf{u}}^{t+1}, s_{bk}(\mathsf{bk}_{\mathsf{u}}^{t+1})) \in \mathbb{L}_{\mathsf{u}}^{t+1}$, then, return $(\mathsf{bk}_{\mathsf{u}}^{t+1}, s_{bk}(\mathsf{bk}_{\mathsf{u}}^{t+1}))$;
- Else if: $|\mathbb{L}_{\mathsf{u}}^{t+1}| > q$ when u is honest or $|U| > Q$, then return $\perp$;
- Else: return $(\mathsf{bk}_{\mathsf{u}}^{t+1}, s_{bk}(\mathsf{bk}_{\mathsf{u}}^{t+1})) \leftarrow \mathsf{US.Sample}(U, (\mathsf{PoBA.Solve}, s_{bk}), (\mathsf{Pool}_{\mathsf{u}}^{t+1}, \mathsf{chain}_{\mathsf{u}}^{t}, r_{\mathsf{u}}^{t+1}))$ *s.t.* $s_{bk}(\mathsf{bk}_{\mathsf{u}}^{t+1}) \overset{\mathcal{D}}{\leftarrow} \mathbb{S}$, and add $(\mathsf{Pool}_{\mathsf{u}}^{t+1}, \mathsf{chain}_{\mathsf{u}}^{t}, r_{\mathsf{u}}^{t}, \mathsf{bk}_{\mathsf{u}}^{t+1}, s_{bk}(\mathsf{bk}_{\mathsf{u}}^{t+1}))$ to $\mathbb{L}_{\mathsf{u}}^{t+1}$.

5.2 Persistence and Block-Liveness

Persistence and liveness [16] are the core properties of our protocol. For persistence, we adopt a refined version from [11]. Whereas for liveness, it is derived from two basic properties called chain growth and chain quality [16]. We observe (1) chain growth follows from our slot-based execution setting (Section 2); and (2) our score-based design eliminates the need of chain quality. That is, the chain quality requires the fraction of adversarial blocks in a time interval to be on par with the fraction of adversarial users. Otherwise, adversaries can exclude typical transactions in their blocks, so the protocol cannot achieve liveness. However, in our case, with a well-designed scoring function (concerning the underlying double auction system), adversaries cannot gain advantages from such an attack as it lowers the block score. Hence, the possibility of the adversarial blocks being selected is lowered. Based on these observations, we define a "block-liveness" property for our protocol instead of the conventional transaction liveness.

Definition 13 (Persistence and Block-Liveness). *Formally, we define:*

– *Persistence: For any two honest users with blockchains* $\mathsf{chain}_1^{t_1}, \mathsf{chain}_2^{t_2}$ *at slot* $t_1, t_2 \geq 0$*, respectively. Without loss of generality, let* $t_1 \leq t_2$*. Persistence with parameter* $\tau \in \mathbb{N}$ *indicates that* $\mathsf{chain}_1^{t_1}$*, is a prefix of* $\mathsf{chain}_2^{t_2}$ *after removing the rightmost* τ *blocks, i.e.,* $\mathsf{chain}_i^{t_1 \lceil \tau} \subseteq \mathsf{chain}_2^{t_2}$*;*
– *Block-liveness: For any honest user with* chain^t *in slot* $t \geq 0$*, block-liveness states that for any* $i \in [t]_0$*,* chain^i *is extended by exact one block.*

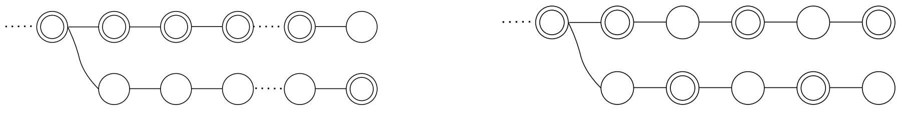
In an honest user's local block-tree, the persistence is proven by: (1) the highest-scored branch of each slot will eventually be known to all honest users; (2) the selected branches of different slots have a long enough common prefix. If a block or branch is known to all honest users, it is disclosed. The δ-bounded network guarantees that honestly generated blocks are always disclosed after δ slots. The adversary is empowered to learn all generated blocks, thereby knowing the master-tree in any given slot. However, we require the adversary to always send the highest-scored blockchain of each slot to at least one honest user. This assumption is made to rule out the possibility of a selfish mining attack, which is particularly challenging to analyze due to its game-theoretic nature [2].

Assumption 1. *Let* $\mathcal{A}$ *be the rushing adversary who holds the master-tree* mtree^t *of slot* $t \geq 0$*. For any* $i \in [t]_0$*, if* $\mathsf{chain}^i = \arg\max_{\mathsf{branch}^i \subseteq \mathsf{mtree}^i} S_{\mathsf{branch}^i}$*, at least one honest user receives* chain^i *by the end of slot* i*.*

Thereby, we have the following Lemma 1 (proof is in Appendix D).

Lemma 1. *Given a* δ*-synchronous network, for any slot* $t \geq 0$*, if* $\mathsf{chain}^t \subseteq \mathsf{mtree}^t$ *is the highest-scored branch in* t*,* chain^t *will be disclosed for any* $\ell \geq t + \delta + 1$*.*

The knowledge of the highest-scored branches of each slot is not enough to prove persistence. Even in the master-tree in which everything is known, given two conjunctive slots, the highest-scored branches may be different from each other, *e.g.*, a high-but-not-highest-scored branch gets extended by an extremely high-scored block so that the new branch is selected in the next slot. This situation causes the blockchain to be unstable and prevents honest users from agreeing on the same blockchain. There are two types of persistence violations: (1) the newly selected branch does not extend previously selected ones so that consensus can get reset (Figure 1a); (2) the change of branch selection frequently happens so that consensus cannot be settled (Figure 1b).



(a) A chain get **substituted** after selected for multiple slots.

(b) Chains **swing** among branches over conjunctive slots.

Fig. 1: Persistence Violations: Substitution and Swing. The circle denotes the blocks on the branches, and the double circle denotes the branch being selected as the highest-scored one.

The first case, we analyze the probability of existing any branch that deviates the selected chain for $\tau \geq 1$ slots getting selected in slot $t+1$. We denote the chain selected in slot t with $\mathsf{chain}^t = \mathsf{chain}^{t\lceil\tau}||\mathsf{bk}_\mathsf{c}^{t-\tau+1}||\ldots||\mathsf{bk}_\mathsf{c}^t$, and *w.l.o.g.*, let $\mathsf{branch}^t = \mathsf{chain}^{t\lceil\tau}||\mathsf{bk}_\mathsf{b}^{t-\tau+1}||\ldots||\mathsf{bk}_\mathsf{b}^t$ be a branch *s.t.* $\mathsf{bk}_\mathsf{c}^i \neq \mathsf{bk}_\mathsf{b}^i$ for all $i \in [t-\tau+1..t]$. The violation indicates that there exists a block $\mathsf{bk}_\mathsf{b}^{t+1}$ that extends branch^t *s.t.* $S_{\mathsf{branch}^t||\mathsf{bk}_\mathsf{b}^{t+1}} > \max_{\mathsf{bk}_\mathsf{c}^{t+1}} S_{\mathsf{chain}^t||\mathsf{bk}_\mathsf{c}^{t+1}}$ where $\mathsf{bk}_\mathsf{c}^{t+1}$ denotes the block candidates extending chain^t. We conclude with Lemma 2 (proof in Appendix E).

Lemma 2. *Let $\mathcal{D}$ be the score distribution determined by $s_{bk}(\cdot)$. Let mtree^{t+1} be the master-tree of any slot $t+1 \geq \tau \geq 1$. Assuming at least one honest user, there exists an accumulating function $acc(\cdot)$ (Definition 11) s.t. the violation case mentioned above occurs with probability less than $\mathsf{O}(c^{-\tau})$ where $c>1$ is a constant value given by $acc(\cdot)$.*

The second violation occurs when the adversary finds the highest-scored *block* in multiple conjunctive slots as honest users stick to the highest-scored branch.

Lemma 3. *Given $\mathcal{D}$ and mtree^{t+1} the same as Lemma 2. Assuming the fraction of adversarial computing power is $f<1$ (i.e., $Q_\mathcal{A} < f \cdot Q$), chain swinging occurs during $\tau \geq 1$ conjunctive slots with probability $\mathsf{O}(f^\tau)$.*

To prove this lemma, we remark that for any distribution, the probability of the adversary finding the highest-scored block is upper bounded by its computing power fraction f. Finally, by Lemma 1 and by applying the Markov chain approach to Lemma 2, 3, we have the theorem for persistence.

Theorem 1 (Persistence). *Let $c>1$ be the constant value given by $acc(\cdot)$ and $f<1$ be the fraction of adversarial computing power, the protocol parameterized by $\tau \geq \delta + 1$ satisfies persistence with probability $1 - \mathsf{O}(c^{-\tau+\delta} + f^{\tau-\delta})$.*

For completeness, we show Theorem 2. The proof is straightforward. Assume the one honest user is unaware of any other blocks. She can trivially extend her block-tree by generating blocks locally and selecting the blockchain accordingly.

Theorem 2 (Block-liveness). *Assuming at least one honest user, the protocol satisfies block-liveness unconditionally.*

6 Implementation

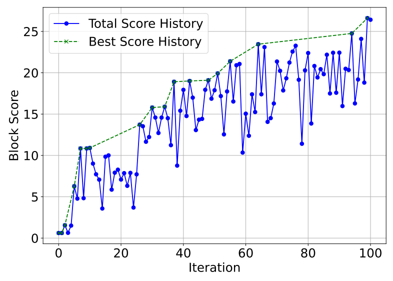
The implementation is for the PoBA scheme using a concrete scoring function: $s_{bk}(\mathsf{bk}) = \sum_{i,j} p_{ij} \cdot q_{ij}$ where $(\mathsf{bidID}_i, \mathsf{bidID}_j, p_{ij}, q_{ij}) \in \mathsf{TXs}$ is the transaction set embedded bk. The code for PoBA and a functional network simulation for our PoBA-based blockchain protocol can be found in the anonymous repository[12].

Setup. Our implementation begins with an SGMAP under the given scoring function, using a bid set of size N=5. We deliberately selected a small N to shorten the solving time, as our primary focus is on validating the output distribution rather than the runtime. To solve this sub-problem, we utilized the SLSQP solver from Python's SciPy package [34]. Next, we prepared two bid sets under different distributions: (1) Bids with uniformly distributed $p \in (0, 1]$ and $q \in [1..10]$; (2) Bids with (normally distributed) $p \sim \mathcal{N}(0.5, 0.01)$ and $q \sim \mathcal{N}(5.5, 4)$, which are further clipped (and rounded) to range $(0, 1]$ (for p) and $[1..10]$ (for q), respectively. Both sets are much larger than N to simulate the bidpool.

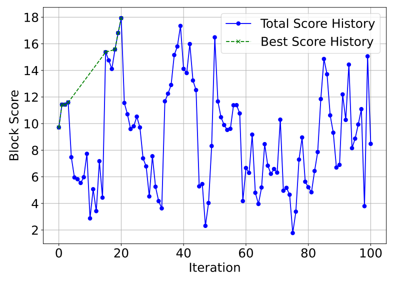
For the optimization process, we utilize two stochastic algorithms in conjunction with the SLSQP solver: simulated annealing (SA) and stochastic local search (SLS). In each iteration, these algorithms explore possible solutions by moving elements in and out of the current sub-problem. To demonstrate the solving process, our simulations run with a maximum of 100 iterations for each algorithm. For large-scale experiments focusing on the score distribution, each algorithm is set to 20 iterations to simulate the limited computing power and shorten the solving time. We conduct the following two series of experiments: (1) Optimization process visualization: Each algorithm (SA and SLS) is run once on the uniformly distributed bids to visualize the score during 100 iterations; and (2) Score distribution analysis: Each algorithm is run 5000 times on both uniform and normal bid distributions, with each run limited to 20 iterations.

Visualization and analysis. The following figures visualize the score history during 100 iterations using SA (Figure 2a) and SLS (Figure 2b). Both results show noticeable fluctuations in block scores throughout the iterations, highlighting the stochastic nature of finding the optimal PoBA solution *w.r.t.* the given scoring function. Consequently, by requiring users in the PoBA scheme to adhere

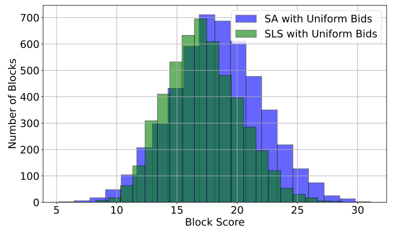
[12] `https://anonymous.4open.science/r/poba-chain`.



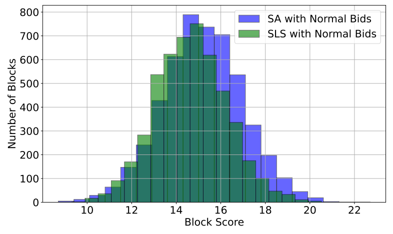
(a) Score history with the SA.



(b) Score history with the SLS.



(c) Uniform bids using SA and SLS.



(d) Normal bids using SA and SLS.

Fig. 2: Optimization process visualization and score distribution analysis.

to a generic enough stochastic solving algorithm, it becomes unlikely that any single user will consistently dominate the block generation.

Next, we illustrate the results of the score distribution in Figure 2c and Figure 2d, comparing different algorithms (SA and SLS) applied to uniformly and normally sampled bids. We consider that our bid set simulates a user's bidpool, which should exhibit some consistency across the network. Although slight shifts are observed in the results, the overall distribution of the best scores remains stable. This stability arises from two key factors: (1) the scores naturally align with the distribution defined by the scoring function; and (2) the resulting distribution is influenced by the generalized extreme value (GEV) distribution (according to the extreme value theory) as each trial involves multiple iterations in our experiments. The stability in our results suggests that, given an input bid distribution, a generic stochastic solving algorithm prevents any user from gaining a significant advantage in finding higher-scored blocks, as the scores are consistently shaped by both the scoring function and the GEV distribution.

The implementation results suggest that we can reliably focus on the score distribution without overly concerned with specific settings, thereby validating our use of the universal sampler model for the generic scoring function.

7 Final Remarks

We proposed a novel blockchain consensus protocol that can be applied to double auction systems, hence having practical applications, *i.e.*, peer-to-peer energy trading. Our construction is somewhat similar to PoW, *i.e.*, relying on computational power. However, it was repurposed for a more useful work, *i.e.*, allocating buy and sell bids. Concretely, we introduced a Scored Generalized Multiple Assignment Problem, SGMAP, and semanticized the problem with our thoroughly redesigned (to accommodate bidding and bid allocation) blockchain data structure. The resulting scheme is proof-of-bid-assignment (PoBA), the core component of our proposed score-based blockchain protocol. Finally, we presented significantly involved security analysis and experimental results to substantiate the security. Concretely, we analyzed honest users' local chain dynamics under our score-based blockchain selection rule to prove persistence; and showed liveness concerning a variant that focuses on block dynamics instead of transactions.

References

1. Badertscher, C., Ga zi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros chronos: Permissionless clock synchronization via proof-of-stake. Cryptology ePrint Archive, Report 2019/838 (2019), `https://eprint.iacr.org/2019/838`
2. Badertscher, C., Garay, J.A., Maurer, U., Tschudi, D., Zikas, V.: But why does it work? A rational protocol design treatment of bitcoin. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 34–65. Springer, Heidelberg (Apr / May 2018). `https://doi.org/10.1007/978-3-319-78375-8_2`
3. Badertscher, C., Lu, Y., Zikas, V.: A rational protocol treatment of 51% attacks. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 3–32. Springer, Heidelberg, Virtual Event (Aug 2021). `https://doi.org/10.1007/978-3-030-84252-9_1`
4. Ball, M., Rosen, A., Sabin, M., Vasudevan, P.N.: Proofs of work from worst-case assumptions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 789–819. Springer, Heidelberg (Aug 2018). `https://doi.org/10.1007/978-3-319-96884-1_26`
5. Blocki, J., Zhou, H.S.: Designing proof of human-work puzzles for cryptocurrency and beyond. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 517–546. Springer, Heidelberg (Oct / Nov 2016). `https://doi.org/10.1007/978-3-662-53644-5_20`
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). `https://doi.org/10.1109/SFCS.2001.959888`
7. Constantinides, T., Cartlidge, J.: Block auction: A general blockchain protocol for privacy-preserving and verifiable periodic double auctions. In: Xiang, Y., Wang, Z., Wang, H., Niemi, V. (eds.) 2021 IEEE Blockchain 2021, Melbourne, Australia, December 6-8, 2021. pp. 513–520. IEEE (2021). `https://doi.org/10.1109/BLOCKCHAIN53845.2021.00078`
8. Daian, P., Pass, R., Shi, E.: Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In: Goldberg, I., Moore, T. (eds.) FC 2019. LNCS, vol. 11598, pp. 23–41. Springer, Heidelberg (Feb 2019). `https://doi.org/10.1007/978-3-030-32101-7_2`
9. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 66–98. Springer, Heidelberg (Apr / May 2018). `https://doi.org/10.1007/978-3-319-78375-8_3`

10. Dyer, M.E., Frieze, A.M.: Probabilistic analysis of the generalised assignment problem. Math. Program. **55**, 169–181 (1992). https://doi.org/10.1007/BF01581197
11. Fitzi, M., Kiayias, A., Panagiotakos, G., Russell, A.: Ofelimos: Combinatorial optimization via proof-of-useful-work - A provably secure blockchain protocol. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 339–369. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_12
12. Galal, H.S., Youssef, A.M.: Verifiable sealed-bid auction on the ethereum blockchain. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) FC Workshop. LNCS, vol. 10958, pp. 265–278. Springer (2018). https://doi.org/10.1007/978-3-662-58820-8_18
13. Galal, H.S., Youssef, A.M.: Trustee: Full privacy preserving vickrey auction on top of ethereum. In: Bracciali, A., Clark, J., Pintore, F., Rønne, P.B., Sala, M. (eds.) FC Workshop. LNCS, vol. 11599, pp. 190–207. Springer (2019). https://doi.org/10.1007/978-3-030-43725-1_14
14. Galal, H.S., Youssef, A.M.: Publicly verifiable and secrecy preserving periodic auctions. In: Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S. (eds.) FC Workshop. LNCS, vol. 12676, pp. 348–363. Springer (2021). https://doi.org/10.1007/978-3-662-63958-0_29
15. Garay, J.A., Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Rational protocol design: Cryptography against incentive-driven adversaries. In: 54th FOCS. pp. 648–657. IEEE Computer Society Press (Oct 2013). https://doi.org/10.1109/FOCS.2013.75
16. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_10
17. Garay, J.A., Kiayias, A., Leonardos, N., Panagiotakos, G.: Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 465–495. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_16
18. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454 (2017), https://eprint.iacr.org/2017/454
19. Górski, T., Bednarski, J.: Modeling of smart contracts in blockchain solution for renewable energy grid. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) 17th Computer Aided Systems Theory - EUROCAST 2019, Las Palmas de Gran Canaria, Spain, February 17-22, 2019, Part I. LNCS, vol. 12013, pp. 507–514. Springer (2019). https://doi.org/10.1007/978-3-030-45093-9_61
20. Hofheinz, D., Jager, T., Khurana, D., Sahai, A., Waters, B., Zhandry, M.: How to generate and use universal samplers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 715–744. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_24
21. Kamp, S.H., Magri, B., Matt, C., Nielsen, J.B., Thomsen, S.E., Tschudi, D.: Weight-based nakamoto-style blockchains. In: Longa, P., Ràfols, C. (eds.) LATINCRYPT 2021. LNCS, vol. 12912, pp. 299–319. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-88238-9_15
22. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978). https://doi.org/10.1145/359545.359563, https://doi.org/10.1145/359545.359563

23. Lamport, L., Shostak, R.E., Pease, M.C.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982). https://doi.org/10.1145/357172.357176, https://doi.org/10.1145/357172.357176
24. Liu, B., Yang, Y., Wang, R., Hong, Y.: Poster: Privacy preserving divisible double auction with A hybridized tee-blockchain system. In: 41st IEEE ICDCS 2021, Washington DC, USA, July 7-10, 2021. pp. 1144–1145. IEEE (2021). https://doi.org/10.1109/ICDCS51616.2021.00128
25. Liu, L., Du, M., Ma, X.: Blockchain-based fair and secure electronic double auction protocol. IEEE Intell. Syst. **35**(3), 31–40 (2020). https://doi.org/10.1109/MIS.2020.2977896
26. Ma, X., Xu, D., Wolter, K.: Blockchain-enabled feedback-based combinatorial double auction for cloud markets. Future Gener. Comput. Syst. **127**, 225–239 (2022). https://doi.org/10.1016/J.FUTURE.2021.09.009
27. Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_6
28. Nguyen, T.D.T., Thai, M.T.: A blockchain-based iterative double auction protocol using multiparty state channels. CoRR (2020), https://arxiv.org/abs/2007.08595
29. Park, J.S., Lim, B.H., Lee, Y.: A lagrangian dual-based branch-and-bound algorithm for the generalized multi-assignment problem. Manage. Sci. **44**(12), 271–275 (dec 1998)
30. Pass, R., Seeman, L., shelat, a.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 643–673. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56614-6_22
31. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 3–33. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_1
32. Roussas, G.: An introduction to probability and statistical inference, pp. 207–243. Elsevier, Academic Press (12 2015). https://doi.org/10.1016/B978-0-12-800114-1.00006-8
33. Thakur, S., Breslin, J.G., Malik, S.: Privacy-preserving energy trade using double auction in blockchain offline channels. In: Prieto, J., Martínez, F.L.B., Ferretti, S., Guardeño, D.A., Nevado-Batalla, P.T. (eds.) 4th IEEE BLOCKCHAIN 2022, L'Aquila, Italy, 13-15 July 2022. Lecture Notes in Networks and Systems, vol. 595, pp. 289–302. Springer (2022). https://doi.org/10.1007/978-3-031-21229-1_27
34. Virtanen, P., et al.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**(3), 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2, https://www.nature.com/articles/s41592-019-0686-2
35. Wongsamerchue, T., Leelasantitham, A.: An electronic double auction of prepaid electricity trading using blockchain technology. J. Mobile Multimedia **18**(6), 1829–1850 (2022). https://doi.org/10.13052/JMM1550-4646.18616
36. Zhang, S., Miao, P., Wang, B., Dong, B.: A privacy protection scheme of microgrid direct electricity transaction based on consortium blockchain and continuous double auction. IEEE Access **7**, 151746–151753 (2019). https://doi.org/10.1109/ACCESS.2019.2946794

A Auxiliary Definitions

This section provides supplementary definitions and descriptions.

A.1 Digital Signatures Scheme

The algorithms in a signature scheme SIG=($\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}$) works as follows.

– $\mathsf{KGen}(1^\lambda)$ takes as input λ and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$;
– $\mathsf{Sign}(\mathsf{sk}, m)$ takes as input the secret key sk and a message m. It outputs a signature σ on m under sk;
– $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$ takes as input the public key pk, the message m and the signature σ. It outputs 1 if the signature is valid and 0 otherwise.

The correctness and the existential unforgeability under adaptive chosen message attacks (EUF-CMA) are defined as follows.

Definition 14 (Correctness). *A signature scheme satisfies correctness, if for any* $\lambda > 0$ *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$*:* $\Pr[\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1] = 1$.

Definition 15 (EUF-CMA). *A signature scheme satisfies EUF-CMA, if for any adversary that has access to a signing oracle* $\mathcal{O}_{\mathsf{Sign}}(\mathsf{sk}, \cdot)$ *with queries* $m \in \mathsf{Q}$*, the following probability is negligible of* λ *for any* $\lambda > 0$ *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$*:*

$$\Pr\left[(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk}) | m^* \notin \mathsf{Q} \wedge \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}) = 1\right].$$

A.2 The Diffuse Functionality

Based on [9, 16], we describe the delayed diffuse functionality tailored to our protocol setting (recall Section 2). The functionality is parameterized by $\delta \in \mathbb{N}$ and denoted by $\mathsf{DDiffuse}_\delta$. It keeps rounds, executing one round per slot. $\mathsf{DDiffuse}_\delta$ interacts with the environment $\mathcal{Z}$, users in slot $t \geq 0$, *i.e.*, u_i for all $i \in [n_t]$, and an adversary $\mathcal{A}$. It works as follows in each round:

– $\mathsf{DDiffuse}_\delta$ maintains a $\mathsf{RECEIVE}(i)$ string defined for each user u_i. A user u_i, if activated, is allowed to fetch the contents of her personal $\mathsf{RECEIVE}(i)$ string at any time. Moreover, users can instruct the functionality to diffuse a potentially empty message m. Activated users are allowed to diffuse once in a round;
– When the adversary is activated, it is allowed to: (1) Read all $\mathsf{RECEIVE}(\cdot)$; (2) Read all diffuse requests and deliver messages to their corresponding $\mathsf{RECEIVE}(\cdot)$ in any order it prefers; (3) For any message m obtained from a diffuse request and any user u_i, $\mathcal{A}$ can move m into a special string $\mathsf{DELAYED}(i)$ instead of $\mathsf{RECEIVE}(i)$;
– At the end of each round, $\mathsf{DDiffuse}_\delta$ ensures that any message to u_i that was either (1) diffused in this round and not put to $\mathsf{DELAYED}(i)$; or (2) removed from the string $\mathsf{DELAYED}(i)$ in this round and delivered to $\mathsf{RECEIVE}(i)$. Moreover, if any message currently present in $\mathsf{DELAYED}(i)$ was originally diffused at least δ slots ago, then $\mathsf{DDiffuse}_\delta$ removes it from $\mathsf{DELAYED}(i)$ and appends it to $\mathsf{RECEIVE}(i)$;
– Upon receiving a create instruction from $\mathcal{Z}$ with a user-tree tree, $\mathsf{DDiffuse}_\delta$ spawns a new user with tree as her local tree and informs her with (n_t, f_t, δ).

B Formal Protocol Description

The PoBA-Based Blockchain Protocol $\Pi^{n,\delta,k,\tau,s_{bk}(\cdot),acc(\cdot)}_{\mathcal{T},\mathsf{PoBA},\mathsf{UpdatePool},\mathsf{UpdateTree}}$

In each slot $t{+}1{\geq}0$ given by the global clock $\mathcal{T}$, the protocol Π is executed by n users (with honest majority). We assume δ is the known network delay and $k{\in}[0,1]$ is a constant that determines the auction price. Given a generic scoring function $s_{bk}(\cdot)$, an accumulating parameter function $acc(\cdot)$, the PoBA scheme PoBA, and the algorithms $(\mathsf{UpdatePool}, \mathsf{UpdateTree})$, instructions for an honest user u with an tuple output from slot t, *i.e.*, $(\mathsf{Pool}^t_{\mathsf{u}'}, \mathsf{tree}^t_{\mathsf{u}'}, B^t_{\mathsf{u}})$, are as follows.

- **Select local blockchain:** $\mathsf{chain}^t {\overset{\Delta}{=}} \arg\max_{\mathsf{branch}^t \subseteq \mathsf{tree}^t_{\mathsf{u}'}} S_{\mathsf{branch}^t}$;
- **Update bidpool:** Run $\mathsf{Pool}^{t+1}_{\mathsf{u}} {\leftarrow} \mathsf{UpdatePool}(\mathsf{Pool}^t_{\mathsf{u}'}, \mathsf{branch}^t, B^t_{\mathsf{u}})$ for each $\mathsf{branch}^t \subseteq \mathsf{tree}^t_{\mathsf{u}'}$ *s.t.* $\mathsf{Pool}^{t+1}_{\mathsf{u}}$ is valid *w.r.t.* chain^t;
- **Generate block candidates:** $\mathsf{chain}^t || \mathsf{bk}^{t+1}_{\mathsf{u},j} {\leftarrow} \mathsf{PoBA.Solve}(\mathsf{chain}^t, \mathsf{Bs}_j)$ where $\mathsf{Bs}_j {\subseteq} \mathsf{Pool}^{t+1}_{\mathsf{u}}$ and $|\mathsf{Bs}_j| {\leq} \mathsf{N}$ *s.t.* $BK^{t+1}_{\mathsf{u}} {\overset{\Delta}{=}} \{\mathsf{bk}^{t+1}_{\mathsf{u},j}\}_{j \in [\mathsf{ite}]}$;
- **Diffuse the best candidate:** Let the best block candidate $\mathsf{bk}^{*t+1}_{\mathsf{u}} {\overset{\Delta}{=}} \arg\max_{\mathsf{bk}^{t+1}_{\mathsf{u},j} \in BK^{t+1}_{\mathsf{u}}} s_{bk}(\mathsf{bk}^{t+1}_{\mathsf{u},j})$ and diffuse $\mathsf{chain}^{*t+1}_{\mathsf{u}} {\overset{\Delta}{=}} \mathsf{chain}^t || \mathsf{bk}^{*t+1}_{\mathsf{u}}$;
- **Update user-tree:** $\mathsf{tree}^{t+1}_{\mathsf{u}} {\leftarrow} \mathsf{tree}^t_{\mathsf{u}'}$, and for each valid chain'^{t+1} received in slot $t{+}1$, $\mathsf{tree}^{t+1}_{\mathsf{u}} {\leftarrow} \mathsf{UpdateTree}(\mathsf{tree}^{t+1}_{\mathsf{u}}, \mathsf{chain}'^{t+1})$;
- **Collect new bids:** Collect valid bids to $B^{t+1} {\overset{\Delta}{=}} \{(\mathsf{bidID{:}bid}) | H(\mathsf{bid}) = \mathsf{bidID} \wedge \mathsf{bid}.t_{\mathsf{Gen}} \leq t{+}1 < \mathsf{bid}.t_{\mathsf{Exp}} \wedge (\mathsf{bidID{:}bid}) \notin \mathsf{Pool}^{t+1}_{\mathsf{u}}\}$;
- **Report confirmed ledger:** When queried by the protocol Π, select $\mathsf{chain}^{t+1}_{\mathsf{u}} {\overset{\Delta}{=}} \arg\max_{\mathsf{branch}^{t+1} \subseteq \mathsf{tree}^{t+1}_{\mathsf{u}}} S_{\mathsf{branch}^{t+1}}$ and output $\mathsf{chain}^{t+1\lceil \tau}_{\mathsf{u}}$.

C Discussion: Incentive Model and Rational Analysis

Our security analysis (Section 5) is based on *assumptions* of the fraction of adversarial users and their behaviors. Note that this work does *not* analyze the reasons behind these assumptions based on users' rationality as in [2, 3, 15]. Instead, we present an intuition of the hardness of our rational analysis here.

Like conventional blockchain protocols, there are two layers of incentive: inherent (*e.g.*, transaction fee) and explicit (*e.g.*, block reward). The inherent incentive in our case derives from where users can tweak transactions to benefit themselves, *e.g.*, assigning higher buy prices for their sell bids or lower sell prices for their sell bids. However, prioritizing their own bids in the solving algorithm will potentially sacrifice the block scores. Hence, their blocks may fail to be selected in the confirmed blockchain. The trade-off between this inherent reward and the scarification of block scores requires a case-by-case analysis with respect to concrete scoring functions.

It is even more complicated to consider explicit incentives. A well-chosen scoring function and our highest-score rule will impose the selected blockchain to benefit the underlying double auction system. Hence, as long as the rushing adversary cannot disturb consensus (proven with respect to persistence and liveness), we regard our protocol as secure. However, in practice, the computation in the PoBA can be unfair so that some users may dominate the block generation. Explicit incentives, like block rewards, will intensify this unfairness.

Moreover, as pointed out by [15], the reason of honest users following protocol instruction is also a crucial aspect for rational analysis, which, unfortunately, is unclear for our protocol[13]. Recall that we inherently assume that users in the PoBA will compete with each other for higher-scored blocks. An instant deficiency caused by replacing honest users with rational agencies is that rational agencies will generate blocks embedding random (but valid) transaction sets to save computing power. Then, even under our highest-score rule, the selected blockchain is far from meaningful for the underlying double auction system.

Therefore, we deliberately leave the design of the incentive model and rational analysis as future work.

D Proof of Lemma 1

We prove Lemma 1 as follows.

Proof. Consider the rightmost block on $\mathsf{chain}^t{=}\mathsf{bk}^G||\ldots||\mathsf{bk}^t$. If bk^t is generated by an honest user, it will be disclosed after δ slots by the network setting. Otherwise, by Assumption 1, at least one honest user receives bk^t in slot t. Because chain^t is the highest-scored branch in mtree^t, and the user's block-tree is a sub-graph of mtree^t, the user will also adopt chain^t as her highest-scored branch of slot t. Then, she (honest) will generate a block atop it in slot $t{+}1$. Therefore, for any $\ell{\geq}t+\delta+1$, chain^t is disclosed. □

E Proof of Lemma 2

We prove Lemma 2 as follows.

Proof of Lemma 2. For $\tau{=}1$, let $\mathsf{branch}_{\mathsf{c}}^{t+1}{=}\mathsf{chain}^{t-1}\ ||\mathsf{bk}_{\mathsf{c}}^{t}||\mathsf{bk}_{\mathsf{c}}^{t+1}$ and $\mathsf{branch}_{\mathsf{b}}^{*t+1}$ $=\mathsf{chain}^{t-1}||\mathsf{bk}_{\mathsf{b}}^{*t}||\mathsf{bk}_{\mathsf{b}}^{*t+1}$ be two branches in slot $t{+}1$ where $\mathsf{chain}^{t-1}, \mathsf{chain}^{t-1}||\mathsf{bk}_{\mathsf{c}}^{t}$ are the selected blockchains in slot $t{-}1$ and t. If a particular $\mathsf{branch}_{\mathsf{b}}^{*t+1}$ substitutes $\mathsf{branch}_{\mathsf{c}}^{t+1}$ (*i.e.*, $\mathsf{branch}_{\mathsf{b}}^{*t+1}$ gets selected), we write the probability with $\Pr[\tau{=}1, \mathsf{branch}_{\mathsf{b}}^{*t+1}]$ as follows (the subscript of the scoring function is omitted).

$$\begin{aligned}\Pr[\tau{=}1, \mathsf{branch}_{\mathsf{b}}^{*t+1}] = \Pr[&s(\mathsf{bk}_{\mathsf{c}}^{t}) - s(\mathsf{bk}_{\mathsf{b}}^{*t}){\geq}0,\\ &c_{0,1}{\cdot}(s(\mathsf{bk}_{\mathsf{c}}^{t}) - s(\mathsf{bk}_{\mathsf{b}}^{*t})) + (s(\mathsf{bk}_{\mathsf{c}}^{t+1}) - s(\mathsf{bk}_{\mathsf{b}}^{*t+1})){\leq}0]\end{aligned} \quad (5)$$

[13] The solution may not be as simple as granting block rewards.

Here, we consider a constant value given by the accumulating function, *i.e.*, $c_{0,1}{=}\frac{acc(t)}{acc(t+1)}$. Denote random variables that represents the scores of the tuple $(\mathsf{bk}_{\mathsf{c}}^{t}, \mathsf{bk}_{\mathsf{b}}^{*t}, \mathsf{bk}_{\mathsf{c}}^{t+1}, \mathsf{bk}_{\mathsf{b}}^{*t+1})$ with $(X_{\mathsf{c}}^{t}, X_{\mathsf{b}}^{t}, X_{\mathsf{c}}^{t+1}, X_{\mathsf{b}}^{t+1})$. Then, write the subtraction of scores with $Y^t{=}X_{\mathsf{c}}^t - X_{\mathsf{b}}^t$ and $Y^{t+1}{=}X_{\mathsf{c}}^{t+1} - X_{\mathsf{b}}^{t+1}$. Following the universal sampler model, $(X_{\mathsf{c}}^{t}, X_{\mathsf{b}}^{t}, X_{\mathsf{c}}^{t+1}, X_{\mathsf{b}}^{t+1})$ are independent and follow the same distribution $\mathcal{D}_X{=}\mathcal{D}$ on $[\mathsf{smin}, \mathsf{smax}]$. Hence, Y^t and Y^{t+1} are independent, and distributed identically and *symmetrically* on $[\mathsf{smin}{-}\mathsf{smax}, \mathsf{smax}{-}\mathsf{smin}]$. We denote the distribution of Y^t and Y^{t+1} with $\mathcal{D}_Y$, and let $f_Y(\cdot)$ and $F_Y(\cdot)$ be the probability density function and the distribution function of $\mathcal{D}_Y$. Therefore, the probability $\Pr[\tau{=}1, \mathsf{branch}_{\mathsf{b}}]$ can be rewritten in the form of random variables

$$\Pr[(Y^t{\geq}0){\wedge}c_{0,1}{\cdot}Y^t + Y^{t+1}{\leq}0)]. \tag{6}$$

Consider the event: $\{Y^t{=}y{\wedge}Y^{t+1}{\leq}{-}c_{0,1}y\}$ for all $y{\in}[0, \mathsf{smax}{-}\mathsf{smin}]$. For simplicity, we rewrite $r{=}\mathsf{smax}{-}\mathsf{smin}$. By Y^t being independent of Y^{t+1}, we have Eq. 6$=\int_0^r \int_{-r}^{-c_{0,1}y} f_Y(y) f_Y(x) \mathrm{d}x\mathrm{d}y$. Then, we can estimate the upper bound with a trick that scales up $f_Y(\cdot)$ with two coefficients $c_1, c_2 > 0$,

$$f_Y(y) \begin{cases} \leq c_1{\cdot}e^{-c_2\cdot y^2}, & \text{if } y \in [-r, r]; \\ =0, & \text{otherwise.} \end{cases} \tag{7}$$

Note that $f_Y(y)$ is a probability density function, hence, it satisfies $\int_{-r}^{r} f_Y(y)$ $\mathrm{d}y{=}1$. Then, for c_1, c_2, we have the estimation: $\int_{-\infty}^{\infty} c_1{\cdot}e^{-c_2\cdot y^2}\mathrm{d}y{\geq}1$, which is $c_1^2/c_2{\geq}\pi^{-1}$. The manipulation of inequality gives us that:

$$\text{Eq. 6}{\leq}\int_0^\infty \int_{-\infty}^{-c_{0,1}y} e^{-y^2}{\cdot}e^{-x^2}\mathrm{d}x\mathrm{d}y{=}\frac{{c_1}^2}{2c_2}{\cdot}\tan^{-1}\left(\frac{1}{c_{0,1}}\right){\leq}\frac{{c_1}^2}{2c_2{\cdot}c_{0,1}}.$$

That is, $\Pr[\tau{=}1, \mathsf{branch}_{\mathsf{b}}^{*t+1}]{\leq}{c_1}^2/(2c_2{\cdot}c_{0,1})$ for any $c_1, c_2{\geq}0, c_1^2/c_2{\geq}\pi^{-1}$ where $c_{0,1}{=}\frac{acc(t)}{acc(t+1)}$.

Next, the probability for any $\mathsf{branch}_{\mathsf{b}}^{t+1}$, denoted by $\Pr[\tau{=}1]$, should consider a joint event, *i.e.*, let q be the number of possible $\mathsf{branch}_{\mathsf{b}}^{t+1}$. Recall that Q is the upper bound of the total number of queries in each slot. Hence, $q{\leq}Q$. Finally, for $\tau{=}1$, we have $\Pr[\tau{=}1]{\leq}1 - \left(1 - \frac{c_1^2}{2c_2\cdot c_{0,1}}\right)^Q$.

The case of $\tau > 1$ follows the same methodology of $\tau{=}1$. Consider a branch that satisfies $\mathsf{branch}_{\mathsf{b}}^{t} \cap \mathsf{chain}^{t} = \mathsf{chain}^{t\lceil\tau}$. Denote the distinct blocks on $\mathsf{branch}_{\mathsf{b}}^{t}$ with $\mathsf{bk}_{\mathsf{b}}^{i}$ for $i{\in}[t{-}\tau{+}1..t]$, and the blocks on chain^t with $\mathsf{bk}_{\mathsf{c}}^{i}$ for $i{\in}[t{-}\tau{+}1..t]$. For the new blocks generated in slot $t{+}1$, denote them with $\mathsf{bk}_{\mathsf{b}}^{t+1}$ for $\mathsf{branch}_{\mathsf{b}}$ and $\mathsf{bk}_{\mathsf{c}}^{t+1}$ for $\mathsf{branch}_{\mathsf{c}}$ (*i.e.*, $\mathsf{chain}^t||\mathsf{bk}_{\mathsf{c}}^{t+1}$ is no longer the highest-scored branch, hence restated as $\mathsf{branch}_{\mathsf{c}}^{t+1}$). By rewriting Eq. 6, the probability of $\mathsf{branch}_{\mathsf{c}}^{t+1}$ substituted by a particular $\mathsf{branch}_{\mathsf{b}}^{*t+1}$, denoted by $\Pr[\tau{>}1, \mathsf{branch}_{\mathsf{b}}^{*t+1}]$, is:

$$\Pr\left[\left(\bigwedge_{i\in[t-\tau+1..t]}\left(\sum_{j=t-\tau+1}^{i}\frac{acc(j)}{acc(i)}{\cdot}Y^j{\geq}0\right)\right)\wedge\left(\sum_{i=t-\tau+1}^{t+1}\frac{acc(i)}{acc(t+1)}{\cdot}Y^i{\leq}0\right)\right]. \tag{8}$$

Here, $Y^i = X_{\mathsf{c}}^i - X_{\mathsf{b}}^i$ for all $i \in [t-\tau+1..t]$ are independent and distributed identically with $\mathcal{D}_Y$ on $[\mathsf{smin}-\mathsf{smax}, \mathsf{smax}-\mathsf{smin}]$. Note that we normalize the coefficient of the last Y in the sum with $\frac{acc(j)}{acc(i)}$ (when $j=i$) and $\frac{acc(i)}{acc(t+1)}$ (when $i=t+1$). For simplicity, we rewrite $\frac{acc(j)}{acc(i)}$ with $c(j,i)$, hence $c(i,i)=1$, and we have ($r_1=0$):

$$\begin{aligned}
\text{Eq. } 8 &= \int_{r_1}^{r} \cdots \int_{r_\tau}^{r} \Pi_{i=t-\tau+1}^{t} \Pr[Y^i = y_i] \cdot \Pr\left[Y^{t+1} \le - \sum_{i=t-\tau+1}^{t} c(i,t+1) \cdot y_i\right] \mathrm{d}y_i^\tau \\
&= \int_{r_1}^{r} \cdots \int_{r_\tau}^{r} \Pi_{i=t-\tau+1}^{t} f_Y(y_i) \cdot F_Y\left(- \sum_{i=t-\tau+1}^{t} c(i,t+1) \cdot y_i\right) \mathrm{d}y_i^\tau \\
&= \int_{r_1}^{r} \cdots \int_{r_\tau}^{r} \int_{-r}^{-\sum_{i=t-\tau+1}^{t} c(i,t+1)\cdot y_i} \Pi_{i=t-\tau+1}^{t} f_Y(y_i) \cdot f_Y(x) \mathrm{d}x \mathrm{d}y_i^\tau. \qquad (9)
\end{aligned}$$

We denote the lower bound of random variable $Y^{t-\tau+i}$ for any $i \in [1..\tau]$ as r_i. Hence $r_i = -\sum_{j=t-\tau+1}^{i} c(j,i) \cdot y_j$. Here, we use a small trick to scale Eq. 9. Note that r_i is not necessarily larger than 0. We scale $y_{t-\tau+i}$ down to $-r$ for all $i \in [1..\tau]$. Then, the upper bound of Y^{t+1} is $Y^{t+1} \le \widetilde{r_{\tau+1}} \overset{\Delta}{=} -c(t-\tau+1, t+1) y_{t-\tau+1} + r \cdot \sum_{j=2}^{\tau} c(t-\tau+j, t+1)$. By $\int_{-r}^{r} f_Y(y_{t-\tau+i}) \mathrm{d}y_{t-\tau+i} \le 1$ for any $i \in [1..\tau]$, we have:

$$\text{Eq. } 9 \le \int_{0}^{r} \int_{-r}^{\widetilde{r_{\tau+1}}} f_Y(y_{t-\tau+1}) f_Y(x) \mathrm{d}x \mathrm{d}y_{t-\tau+1}. \qquad (10)$$

Finally, utilizing the same trick for $f_Y(\cdot)$ as in Eq. 7, we have:

$$\text{Eq. } 10 \le = \frac{{c_1}^2}{c_2} \cdot \tan^{-1}\left(\frac{1}{c(t-\tau+1, t+1) - \sum_{j=2}^{\tau} c(t-\tau+j, t+1)}\right)$$

In order to obtain a similar result as the case of $\tau = 1$, *i.e.*, $\Pr[\tau > 1, \mathsf{branch}_{\mathsf{b}}^{*t+1}] \le \frac{{c_1}^2}{c_2 \cdot c(t-\tau+1, t+1)}$, we need to further scale the sum formula to a higher-order infinitesimal of $c(t-\tau+1, t+1)$, which can be achieved when setting $acc(t) = c^t$ and $c > 2$. In this situation, for any $c_1, c_2 \ge 0$ and $c_1^2/c_2 \ge \pi^{-1}$, the probability for any $\mathsf{branch}_{\mathsf{b}}^{t+1}$ concerning our universal sampler model is:

$$\Pr[\tau > 1] \le 1 - \left(1 - \frac{c_1^2}{c_2 \cdot c(t-\tau+1, t+1)}\right)^Q$$

Combining the discussion above, we consider a constant value $c > \max\{2, \frac{c_1^2}{c_2}\}$ for $acc(t) = c^t$. Then, $\Pr[\tau \ge 1] \le 1 - \left(1 - \frac{c_1^2}{c_2 \cdot c^{-\tau}}\right)^Q$ is of the same order as $Q \cdot c^{-\tau}$. Since we only use the upper bound of total queries to the universal sampler instead of honest queries, our conclusion holds even there is at least one honest user. Finally, we conclude the first violation case occurs with probability $\mathsf{O}(c^{-\tau})$. $\square$

F PoBA, but in the Weight-Based Framework

The core idea of blockchain consensus based on proof-of-work (PoW) schemes is that valid solutions to the PoW puzzle (*w.r.t.* a difficulty threshold) are hard to find. Hence, blocks embedding such solutions are sparse, ensuring that each valid block has enough time (longer than the network delay) to be finalized. The average time between blocks, determined by the threshold, is referred to as the *block time*, which must be set to a worst-case value to prevent the situation where there are unfinalized blocks. However, this setting significantly burdens the network throughput.

To tackle the issue above, a weight-based PoW framework [21] is proposed, where solutions are assigned continuous weights instead of binary ones: 0 when below the threshold; or 1 when meeting the threshold. Similar to the conventional PoW, it remains hard to find a heavy solution. The trick is that, users can submit solutions that do not surpass the threshold when finding one takes too long, offering them a chance to be selected and extend the blockchain.

In particular, the authors proposed a "capped exponential weight function" that amplifies the weight of each solution (hence, block) exponentially based on its relevance with the threshold. Although this leads to a different consensus mechanism, the exponentiation shares a similar design philosophy with the accumulating function used in our highest-score-based blockchain selection, where settled blocks on the chain receive an exponential amplification of their scores.

Considering that: (1) our analysis focuses on the score distribution of the scoring function; and (2) there are transformation functions [32] that can modify the distribution of random variables. Therefore, it raises the following question:

Can we integrate PoBA into the weight-based consensus framework by transforming the scoring function?

Concretely, given a hash function $H : \{0,1\} \to \{0,1\}^\lambda$, a threshold T, and a constant value c (which controls the increase rate of the weight function, *i.e.*, a higher c makes the function closer to the form of binary weights), the normalized capped exponential weight function $w : \{0,1\}^\lambda \to [0,1]$ is defined as follows.

$$w(h) \triangleq \begin{cases} e^{(h-T-1)c}, & \text{if } h \leq T; \\ 1, & \text{otherwise.} \end{cases}$$

Thus, the probability density function of the weight distribution $\mathcal{D}_w$, defined by $w(\cdot)$, can be expressed as $\Pr[X = h] = w(h)$ where X is the random variable representing the weight of hash values.

Note that the part on the left of the threshold follows an exponential distribution. Then, the problem above can be divided into two parts: For any scoring function, (1) is there a transformation function that can modify the score distribution to an exponential distribution? (2) how should the threshold in the PoBA scheme (*w.r.t.* the scoring function) be determined?

Unfortunately, there is no generic transformation that can convert arbitrary distributions into an exact exponential distribution. However, by applying inverse transformations, such as $Y = 1/|X|$, we can generate approximately exponential-like distributions from various baseline distributions, such as uniform (which results in an exact exponential distribution) and normal distributions. Further research is required to assess the applicability of these exponential-like distributions within the context of a weight-based consensus framework.

As for the second question regarding how to determine the threshold in the PoBA scheme for a given scoring function, this remains challenging. Unlike hash functions, which have a clearly defined range, $\{0,1\}^\lambda$, the range of a scoring function depends heavily on its input set, making it difficult to choose an appropriate threshold in advance. However, we suggest that for scoring functions with simpler structures, it may be possible to establish a reasonable threshold based on the properties of the input set.