Peer-to-Peer Energy Trading on Blockchain: Consensus from Bid-Assignment-Based Proof-of-X

Anonymous

***Abstract*—The demand for peer-to-peer energy trading (P2PET) grows alongside the advancement of smart grids. Similar to a double-sided auction market, a P2PET system enables its participants to trade energy by issuing bids/asks to buy/sell energy. A robust ledger that satisfies persistence and liveness, *e.g.*, the Bitcoin Blockchain (EuroCrypt '15), can help to record trading agreements, *i.e.*, transactions combining bids and asks. However, existing blockchain-based P2PET approaches rely on general-purpose blockchains with smart contract capabilities, unavoidably incurring in high operational costs. This work designs a P2PET tailored blockchain based ledger protocol by, first, revisiting the blockchain data structure to support bidding operations. Then, abstracting the process of forming transactions with a Bid-Assignment Problem (BAP), which is a scored variant of the generalized multiple assignment problem. Then leveraging the score-based problem, we propose a proof-of-X (PoX) scheme, to design the corresponding protocol. The key difference from any previous work is that our protocol selects blocks according to their content, *e.g.*, the score of bids/asks and transactions. Hence, a higher-scored block would be preferable to the underlying P2PET system regardless of whether the block's creator is honest. This is because, intuitively, such a block increases the quality (defined by the score) of trading agreements. Finally, under the universal sampler model (AsiaCrypt '16), we prove the security of our blockchain design from the BAP-based PoX.**

***Index Terms*—Peer-to-Peer Energy Trading, Blockchain Protocol, Proof-of-X, Many-to-Many Assignment Problem**

I. Introduction

A. Background

A smart grid involves prosumers who both produce and consume energy, with the infrastructure providing users with equipment to produce, store, and transmit energy. A control system, *i.e.*, the P2PET system, monitors and manages users' operations within the smart grid. P2PET users are equipped with certified, *i.e.*, tamper-proof, smart-meters that measure energy production and consumption. Each one with a Home Energy Management System (HEMS) for energy management. HEMS are more sophisticated, *e.g.*, being capable of computation, and are not assumed to be trustworthy. Similar to combining several local smart grids into a regional grid, the standalone P2PET systems of these grids can also join together to form more extensive systems. Such a process can be done recursively under a hierarchical structure.

This work focus on a standalone small community P2PET system where users and a power plant are connected via bidirectional power lines in a fixed physical topology. Due to the unreliability of energy production, users may face energy shortages or excesses. However, the capability of storing and transmitting energy can stabilize the community's overall energy demands by enabling users to trade with each other, *i.e.*, to buy/sell energy. These operations resemble double-sided auction market: users issue *bids* to buy something (energy in the P2PET's case) at a price, or *asks* to sell something at a price. A trading agreement forms when a *bid* and an *ask* satisfy the condition where the *bid*'s buy price is higher than the *ask*'s sell price. Then, users transmit energy and pay according to the agreement. Therefore, the paramount functionality of P2PET is to securely record the trading agreements history. In a decentralized environment, a public ledger backed by user consensus is used for this purpose, *i.e.*, to prevent any party from tampering with its data.

B. Related Works and Motivation

Due to the popularity of Bitcoin [1] and other cryptocurrencies, consensus and its embodiment as a blockchain protocol has gained a much wider interest, taking the form of a distributed ledger, ensuring persistence and liveness [2]. Hence, it is natural to utilize a blockchain to implement a distributed ledger in P2PET. Numerous studies investigate distributed P2PET [3]–[5]. However, these studies are built atop general-purpose blockchain protocols and rely on smart contract capabilities [6].Unfortunately, this severely limits the opportunities for optimizations, and these protocols suffer from high maintenance fees [7], where users must incur severe costs while submitting smart contract-based transactions to the network. Moreover, the mining mechanism relies on the hash-based proof-of-work (PoW), which requires significant computational power (and hence energy), a sharp contrast with the goals of improving energy efficiency and reducing energy waste. Needless to say, such repeated evaluations, as a computational problem, have no connection with the market that may exist atop the system.

This work explores the design of a dedicated blockchain protocol for P2PET, purposely deviating from smart contracts and PoW. We thoroughly redesign the blockchain data structure itself to exploit the operations in the P2PET system and integrate them into a novel assignment problem-based proof-of-X (PoX) approach named *Proof-of-Bid-Assignment (PoBA)*. Unlike the PoW paradigm, in which much computing (and electrical) power is used to block generation for ensuring a robust ledger, ours leverages the existing architecture and structure of power distribution to provide a leaner and more effective solution. We emphasize that this work focuses on the design of the blockchain protocol instead of the P2PET system or smart grids. Thus we will *not* fully investigate the dynamics of the energy trading market or problems on the infrastructure level, *e.g.*, energy transmission loss.

C. Our Approach and Contributions

We start with refining the blockchain data structure to support bidding operations (We will use "sell bids" instead of "asks" in the following of this paper). Thus, trading agreements, *i.e.*, transactions, are combinations of buy and sell bids. Then, we formalize the process of forming transactions with a many-to-many assignment problem concerning a general scoring function that quantifies the quality of solutions. The problem is defined as the bid-assignment problem (BAP), which is a scored variant of the generalized multiple assignment problems (GMAP) [8]. (**Section III**)

Next, based on the BAP, we propose the formal syntax of the PoBA scheme, in which each user maintains a bidpool, solves the BAP, and evaluates solutions according to the given scoring function. Since we will not fully investigate the underlying market and want to showcase the flexibility of our design, we leave the scoring function general. Therefore, to analyze the computation in PoBA concerning a general scoring function, we consider the universal sampler model [9], [10] that focuses on output (score) distribution instead of concrete functions. This approach is similar to modeling the hash computation in PoW with random oracle [2]. (**Section IV**)

A key difference between our BAP-Based PoX (PoBA) and existing optimization problem-based proof-of-useful-work schemes [11], [12] is that generating *valid* blocks in PoBA does not require users to contribute much computing power. Instead, we consider a novel block/chain selection rule, the "highest-score" rule, which utilizes the scoring function. Concretely, users must compete with each other using their block candidates because we require them to select only the highest-scored one to extend the blockchain. The advantage of this approach is that we can eliminate the requirement of honest block's fraction (*i.e.*, chain quality defined by [2], to be explained in detail in Section V-C), hence, achieving liveness more straightforwardly. Moreover, for persistence, we consider a block-tree structure for our new chain selection rule (because there will be multiple valid blocks within each time interval). We prove that persistence holds for our protocol, assuming any arbitrary score distribution under the universal sampler model. The security proof of persistence is quite involving, and we consider it a significant technical contribution. (**Section V**)

Our contribution is threefold: (1) we formalize the process of transaction formation in P2PET with the BAP, which is a scored variant of GMAP; (2) we propose a PoX scheme based on the BAP, *i.e.*, PoBA; (3) based on the PoBA and utilizing a novel chain selection rule, *i.e.*, the highest-score rule, we propose a provably secure (concerning persistence and liveness) blockchain protocol for P2PET (which may also work for general double-sided auction markets).

II. Preliminaries

Throughout this paper, we use $x \overset{\$}{\leftarrow} X$ to denote x being uniformly and randomly sampled from set X. When specifying distribution, $x \overset{\mathcal{D}}{\leftarrow} X$ denotes that x is randomly sampled from X following distribution $\mathcal{D}$. For an algorithm Alg, $x \leftarrow \mathsf{Alg}$ denotes that x is assigned the output of the algorithm Alg on fresh randomness. For $k \in \mathbb{N}$, let $[k] \overset{\Delta}{=} \{1, \dots, k\}$, while $(\mathsf{key}{:}\mathsf{value})$ denotes a mapping from a key key to its corresponding value value. For completeness, we employ a digital signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ that satisfies correctness and existential unforgeability under adaptive chosen message attacks ($\mathsf{EUF\text{-}CMA}$) [13]. We also use a collision-free hash function $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^\lambda$.

The protocol execution model will be described as follows.

A. Protocol Execution Model

We follow the standard Interactive Turing Machines (ITM) Model approach [14]. 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. We consider the following settings in this work.

- Time slots: Time is divided into discrete units called time slots, indexed by an integer $t \in \{1, 2, \dots\}$. We assume a (not necessarily full) synchronized clock $\mathcal{T}$, equipped with a key pair $(\mathsf{sk}_\mathcal{T}, \mathsf{pk}_\mathcal{T})$ from the signature scheme SIG. Users can submit queries $(\sigma_\mathcal{T}, t) \leftarrow \mathcal{T}(m)$ such that $(\sigma_\mathcal{T}, t) \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathcal{T}, m, t)$ where m is the query message and t is the index of the current slot;
- Synchrony: We adapt the δ-synchronous setting from [14] to our slot-based execution where δ is the known network delay. Suppose an honest user sends a message in slot t. The message is guaranteed to be known to all honest users in any slot $\ell \geq t + \delta$. We assume the diffusion model [2];
- Rushing adversary: We consider a rushing network adversary who can: (1) receive any message from honest users 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, to show the security of our protocol design, we add another limitation to the adversary (formalized with Assumption 1 in Section V-C);
- Permissionless setting with static corruption: We follow the constrained permissionless setting from [15]. In each slot t, exactly $n \in \mathbb{N}$ users execute the protocol with corruption fraction f (*i.e.*, $f{\cdot}n$ corrupted nodes). Whenever an honest user joins, the protocol informs the adversary the parameters (n, δ). We assume a static corruption model in which honest users cannot be corrupted after spawning.

Remark (P2PET Rationale): A P2PET system requires users to perform energy transmission and payment periodically in real life. Hence, time is composed of time slots. Local clocks of P2PET users are implemented by certified smart-meters, which may not be fully synchronized in real life. However, we can adjust the length of time represented by a time slot long enough to make any discrepancies between users' local time insignificant, which is convenient to assume the existence of a synchronized clock. Moreover, the certified hardware naturally enables us to consider a *permissioned setting*, in which a constant number of users are initialized before the protocol execution and are informed with the identities of all honest ones. However, such a setting is not necessary for our security proof. Therefore, we adopt the constrained permissionless setting as shown above.

III. Preparation for P2PET on Blockchain

We first show the refined blockchain data structure that supports bidding operations as in Section I-A. Then, we introduce a scored variant of the generalized multiple assignment problem (GMAP) [8] and integrate it with the refined blockchain data structure. Our proposed problems capture the process of assigning buy/sell bids.

A. Refined Blockchain Data Structure

Because buyers/sellers issue bids to buy/sell energy, we add a bid layer to the conventional "transaction-block" blockchain data structure to support these operations. Each transaction is a trading agreement, which is the energy transmission and payment agreement between a buyer and a seller. Recall the hash function $\mathsf{Hash}(\cdot)$ and the secure signature scheme $\mathsf{SIG}{=}(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ [13] as in Section II.

Each bid has two options: (1) to buy a quantity of energy units for an initial price (anything lower is acceptable); (2) to sell a quantity of energy units for an initial price (anything higher is acceptable). Concerning validity, a bid should also involve its issuer's signature and generation/expiration slots signed by the time server. Then, a transaction is defined as the combination of a buy and a sell bid at a selected quantity and price. Note that our protocol can handle complex bid assignments. Hence, we leave concrete restrictions of transaction validity to Section IV. Next, a block embeds a set of bids and transactions derived from the bid set. For validity, a block should contain a hash link pointing to its parent block. Similar to bids, a block should involve its creator's signature and the generation slot signed by the time server.

Definition 1 ((Simplified) Bid, Transaction, Block):

- Bid: A bid, $\mathsf{bid}{=}(\mathsf{type}, q, p, \mathsf{aux})$, has an identifier $\mathsf{bidID}{=}\mathsf{Hash}(\mathsf{bid})$ where: $\mathsf{type}{\in}\{\mathsf{buy}, \mathsf{sell}\}$ is the bid's type, *i.e.*, a buy bid or a sell bid; q, p denote the bid's quantity and unit price, respectively; aux contains the bid's generation/expiration slots $(t_{\mathsf{Gen}}, t_{\mathsf{Exp}})$ alongside the user's and the time server's signatures. The bid space is denoted by $\mathbb{BID}$;
- Transaction: A transaction is defined as $\mathsf{tx}{=}(\mathsf{bid}_1, \mathsf{bid}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}})$[1] in transaction space $\mathbb{TX}$, with an identifier $\mathsf{txID}{=}\mathsf{Hash}(\mathsf{tx})$ where: $\mathsf{bid}_1, \mathsf{bid}_2$ are two bids of different types; $q_{\mathsf{tx}}, p_{\mathsf{tx}}$ denotes the agreed quantity and unit price.
- Block: A block is defined as $\mathsf{bk}{=}(\mathsf{prevHash}, \mathsf{BIDs}, \mathsf{TXs}, \mathsf{aux})$ and is associated with an identifier $\mathsf{bkID}{=}\mathsf{Hash}(\mathsf{bk})$ where: $\mathsf{prevHash}$ is the identifier of the block that bk extends; $\mathsf{BIDs}, \mathsf{TXs}$ denote the set of bids and transactions embedded in the block; aux contains the block's generation slot t_{Gen} alongside the user's and the time server's signatures. The block space is denoted by $\mathbb{BK}$.

Finally, we define the chain as an ordered linked list of blocks following conventional blockchain protocols. The first block is called *genesis block* and denoted by bk^G, which contains the public keys of initial users.

[1]In practice, using identifiers, bidID_i $(i{\in}\{1,2\})$, instead of bid_i, can save storage for the system.

Definition 2 (Chain): Denote block concatenation with $||$, *i.e.*, if $\mathsf{bk}^i||\mathsf{bk}^{i+1}$, then $\mathsf{prevHash}{\in}\mathsf{bk}^{i+1}$ equals to $\mathsf{Hash}(\mathsf{bk}^i)$, the identifier of bk^i. Hence, a chain in slot $t{\geq}1$ is defined as an ordered linked list of blocks: $\mathsf{chain}^t{=}\mathsf{bk}^G||\mathsf{bk}^1||\mathsf{bk}^2||\cdots||\mathsf{bk}^t$ where bk^G is the genesis block.

Note that our protocol considers a tree structure (which will be explained in Section V). Hence, we may also use "branch", denoted by branch (given in Definition 10), as an interchangeable term of "chain".

B. From Scored GMAP to Bid-Assignment Problem

Now, we consider the assignment of bids, which resembles the formation of transactions. To capture this process, we propose a scored variant of the generalized multiple assignment problem (GMAP) [8]. The scored GMAP is generic and will be semanticized in terms of our refined blockchain data structure by the end of this section.

Starting with inputs, we denote them with two disjoint sets $B{=}\{(q_i^B, p_i^B)\}_{i\in[m]}$ and $S{=}\{(q_j^S, p_j^S)\}_{j\in[n]}$ where $m, n{\in}\mathbb{N}$. Hence, all combinations between B and S is given by the index space $\mathbb{I}{=}\{(i,j){:}\forall i{\in}[m], j{\in}[n]\}$. Then, each assignment between elements from the two sets can be defined by a tuple with respect to indices: $=(i, j, q_{ij}, p_{ij})$ where q_{ij}, p_{ij} are the assigned values. Given an input $B{\cup}S$, we denote an assignment set with $A_{B\cup S, I}{=}\{(i, j, q_{ij}, p_{ij}){:}(i,j){\in}I\}$ where $I \subseteq \mathbb{I}$ is an index set. Later, we will omit inputs $B{\cup}S$ when causing no ambiguity. Therefore, the space of assignment sets given by $B{\cup}S$ can be written as $\mathbb{A}_{B\cup S}{=}\{A_I{:}\forall I \subseteq \mathbb{I}\}$.

Next, we consider constraints concerning the underlying bid assignment process. Let q and p represent quantity and price in bids, respectively. Hence, q_{ij} and p_{ij} can be considered as the assigned quantity and price within transactions. Then, the total amount of the assigned quantity of a bid should be less than the bid's original quantity. Whereas, the assigned price of a meaningful trading agreement should lie between the sell price and the buy price of the original bids. These constraints resemble a many-to-many assignment problem, the GMAP [8]. With the notations mentioned above, the constraints are:

$$q_{ij} \geq 0, \ \sum_{j=0}^{n-1} q_{ij} \leq q_i^B, \ \sum_{i=0}^{m-1} q_{ij} \leq q_j^S;$$
$$p_j^S \leq p_{ij} \leq p_i^B, \text{ if } q_{ij} \neq 0. \tag{1}$$

Finally, the scoring function that evaluates a given assignment set with a real value is denoted by $s : \mathbb{A}{\to}\mathbb{R}$ where $\mathbb{A}{=}\{\mathbb{A}_{B\cup S}{:}\forall B, S\}$. We integrate the scored GMAP with our blockchain, which is defined as bid-assignment problem (BAP), by rewriting B, S with the sets of buy bids and sell bids, respectively, *i.e.*, $B{=}\{\mathsf{bid}_i^B{=}(\mathsf{buy}, q_i^B, p_i^B)\}_{i\in[m]}$, $S{=}\{\mathsf{bid}_j^S{=}(\mathsf{sell}, q_j^S, p_j^S)\}_{j\in[n]}$, with $\mathsf{BIDs}{=}B{\cup}S$.

Further, we rewrite the assignment set into a transaction set $\mathsf{TXs}_I{=}\{\mathsf{tx}_{ij}{=}(\mathsf{bid}_i^B, \mathsf{bid}_j^S, q_{ij}, p_{ij}){:}(i,j){\in}I\}$. Then, the scoring function of transaction sets is redefined as $s_{\text{txs}} : \mathbb{TX}^*{\to}\mathbb{R}$. To extend the scoring function so that it evaluates blocks instead of transaction sets, we first consider the other content of a block as an auxiliary (string), *i.e.*,

3

given a block $\mathsf{bk}{=}(\mathsf{prevHash}, \mathsf{BIDs}, \mathsf{TXs}, \mathsf{aux})$, we denote $(\mathsf{prevHash}, \mathsf{BIDs}, \mathsf{aux}){\in}\{0,1\}^*$ with AUX. Write the scoring function for auxiliary strings with $s_{\text{aux}}{:}\{0,1\}^*{\to}\mathbb{R}$ and let $\mathsf{Agg}_{\mathsf{bk}} : \mathbb{R}^2{\to}\mathbb{R}$ be a function that aggregates two real value scores. Finally, the scoring function of blocks can be given by $s_{\mathsf{bk}} : \mathbb{BK}{\to}\mathbb{R}$ such that for any block $\mathsf{bk}{=}(\mathsf{TXs}, \mathsf{AUX})$:

$$s_{\mathsf{bk}}(\mathsf{bk}){=}\mathsf{Agg}_{\mathsf{bk}}(s_{\text{txs}}(\mathsf{TXs}), s_{\text{aux}}(\mathsf{AUX})). \tag{2}$$

The formal definition of BAP is shown as follows.

Definition 3 (BAP): Let $\mathsf{BIDs}{=}B{\cup}S$ be a set of bids where $B{=}\{\mathsf{bid}_i^B{=}(\mathsf{buy}, q_i^B, p_i^B)\}_{i\in[m]}$ is a buy bid set, and $S{=}\{\mathsf{bid}_j^S{=}(\mathsf{sell}, q_j^S, p_j^S)\}_{j\in[n]}$ is a sell bid set with $m, n{\in}\mathbb{N}$. Let $\mathbb{I}{=}\{(i,j){:}\forall i{\in}[m], j{\in}[n]\}$ be the index space. Let $s_{\mathsf{bk}}{:}\mathbb{BK}{\to}\mathbb{R}$ be a scoring function of blocks defined by Equation 2. The BAP is to find a block bk that maximizes $s_{\mathsf{bk}}(\mathsf{bk})$, and the transaction set $\mathsf{TXs}{\in}\mathsf{bk}$, *i.e.*, $\{\mathsf{tx}_{ij}{=}(\mathsf{bid}_i^B, \mathsf{bid}_j^S, q_{ij}, p_{ij}){:}(i,j){\in}I \subseteq \mathbb{I}\}$ satisfies the constraints given by Equation 1.

Note that it is easy to find a valid solution to a given scored GMAP (or the BAP) as any random assignment set that satisfies constraints in Equation 1 is sufficient. In contrast, the difficulty of finding an optimal solution depends on the scoring function. However, as proven in [8], the problem is NP-complete even when the scoring function is a linear combination of q_{ij} for $i{\in}[m], j{\in}[n]$. Hence, this work will leave scoring functions ($s(\cdot)$ and $s_{\mathsf{bk}}(\cdot)$) general. This choice will also showcase the flexibility of our design.

Remark (Real-Life Extensions): Our data structure and BAP can be extended according to real-life requirements in the P2PET system. For example, users may have preference targets to buy or sell. Hence, we can embed a target list (ordered according to priority) within the bid data structure. Then, the BAP should: (1) consider the target constraints in addition to constraints in Equation 1; (2) adjust the scoring function so that prioritized targets are granted higher scores. Moreover, the scoring function in BAP can be twisted to incentivize provers to include typical bids or transactions into their blocks. For example, the scoring function can grant higher scores to residual bids when the amount of unassigned bids rises (so that those users' energy demands can be fulfilled). We cannot list all possible extensions as real-life demands vary among system instantiations. However, these examples demonstrate our abstraction's capability of modeling real-life systems.

IV. Proof-of-Bid-Assignment (PoBA) Scheme

Like conventional proof-of-X (PoX) schemes, our PoBA involves two types of participants: provers and verifiers, which can be performed by the same user in the protocol. The separation here only aims to clarify the algorithms: provers solve a BAP to generate blocks; whereas, verifiers evaluate solutions' validity and scores.

For concrete settings, let each user to maintain a pool of bids with its view of the blockchain. The "bidpool" is similar to the mempool in other blockchain protocols, with the difference that the mempool keeps transactions. A bidpool should be updated in each time slot according to the bidding history of the P2PET system and the transaction history recorded by the blockchain. Since this work focuses on the assignment process, which involves *quantity of energy* and *price per unit*, we assume any given bid is "alive", *i.e.*, the current slot lies within the bid's generation and expiration slots.

A. History and Bidpool Update

A user's bidpool has a view of all "available" bids in each time slot, which should (1) include newly issued bids in the previous slot; (2) extract assigned bids[2] embedded in the user's blockchain. Let $t{\geq}1$ be the current slot. In the first step, let the set of bids issued in slot $t{-}1$ be $P^{t-1}{=}\{\mathsf{bid}{:}t_{\mathsf{Gen}}{=}t{-}1\}$ where t_{Gen} is each bid's generation slot.

The second operation requires to track the history of bids and transactions. In the following, for simplicity, we only consider the bid set and transaction set in blocks, *i.e.*, we rewrite blocks as $\mathsf{bk}{=}(\mathsf{BIDs}, \mathsf{TXs})$.

Definition 4 (Bid and Transaction History): For any user, let $\mathsf{chain}^t{=}\mathsf{bk}^G||\mathsf{bk}^1||\ldots||\mathsf{bk}^t$ be her blockchain where $t{\geq}1$. For any $i{\in}[t]$, consider $(\mathsf{BIDs}^i, \mathsf{TXs}^i) \in \mathsf{bk}^i$. The bid and transaction history with respect to chain^t is defined over identifiers as $\boldsymbol{H}_{\mathsf{bid}}^t{=}\{\mathsf{bidID}{:}\mathsf{bid}{\in}\bigcup_{i=1}^t \mathsf{BIDs}^i\}$ and $\boldsymbol{H}_{\mathsf{tx}}^t{=}\{\mathsf{txID}{:}\mathsf{tx}{\in}\bigcup_{i=1}^t \mathsf{TXs}^i\}$ where bidID and txID are the identifiers of bid and tx, respectively. We may also use the corresponding bid or transaction for given identifiers.

Now, consider the situation that a given bid's quantity is not fully assigned in a slot. To reduce waste in energy trading, we enable provers to use the residual quantity of that bid in later slots. We formalize this process with the definition of residual bids. Intuitively, a residual bid is a bid that has its identifier existing in the bid history and has an unassigned quantity.

Definition 5 (Residual Bid): Let $\mathsf{chain}^t{=}\mathsf{bk}^G||\mathsf{bk}^1||\ldots||\mathsf{bk}^t$ be a blockchain of slot $t{\geq}1$. Denote its bid and transaction history with $\boldsymbol{H}_{\mathsf{bid}}^t$ and $\boldsymbol{H}_{\mathsf{tx}}^t$. Given a bid $\mathsf{bid}{=}(\mathsf{type}, q, p, \mathsf{aux})$ with identifier $\mathsf{bidID}{\in}\boldsymbol{H}_{\mathsf{bid}}^t$, the residual bid is defined as $\mathsf{rbid}{=}(\mathsf{type}, q_{\mathsf{rbid}}, p, \mathsf{aux})$ with identifier bidID if $q_{\mathsf{rbid}}{>}0$, and $\mathsf{rbid}{=}\perp$ if $q_{\mathsf{rbid}} \leq 0$. Here:

$$q_{\mathsf{rbid}}{=}q - \sum_{\{\mathsf{tx}:\mathsf{txID}\in\boldsymbol{H}_{\mathsf{tx}}^t\wedge\mathsf{bidID}\in\mathsf{tx}\}} q_{\mathsf{tx}}. \tag{3}$$

Provers can assign residual bids into new transactions without exhausting their q_{rbid}, thereby deriving a new residual bid from the original residual bid. Hence, provers can assign bids recursively as long as the bid is alive. Since the residual bid's identifier is identical to its original (residual) one, it is possible to maintain the history of each bid by only tracking unique identifiers. We denote bidpools by Pool and map its elements: $(\mathsf{bidID}{:}\mathsf{bid}\text{ or }\mathsf{rbid}){\in}\mathsf{Pool}$ where bidID is the identifier of (residual) bids. We formally specify the bidpool update algorithm $\mathsf{UpdatePool}$, and show the definition of bidpool validity: Algorithm 1.

Definition 6 (Validity of Bidpools): Let Pool^t be a user's bidpool in slot $t{\geq}1$, and let $\mathsf{chain}^{t-1}{=}\mathsf{bk}^G||\mathsf{bk}^1||\ldots||\mathsf{bk}^{t-1}$ be the valid blockchain in her view (to be defined in Definition 8).

[2]Precisely, assigned quantity within bids. Details can be found in Definition 5.

Let $\boldsymbol{H}^{t-1}_{\mathsf{bid}}$ and $\boldsymbol{H}^{t-1}_{\mathsf{tx}}$ denote the history of chain^{t-1} with $\boldsymbol{H}^{t-1}_{\mathsf{bid}}$ and $\boldsymbol{H}^{t-1}_{\mathsf{tx}}$, respectively. For each $\mathsf{bidID}{\in}\mathsf{Pool}^t$, let its corresponding (residual) bid have quantity q_{rbid}, and have generation and expiration slots $(t_{\mathsf{Gen}}, t_{\mathsf{Exp}})$. Pool is valid with respect to chain if the following conditions hold: (1) The current slot $t{\in}[t_{\mathsf{Gen}}, t_{\mathsf{Exp}}]$; (2) For any $\mathsf{bidID}{\in}\boldsymbol{H}^{t-1}_{\mathsf{bid}}$, q_{rbid} computed from Equation 3 is larger than 0; (3) For any $\mathsf{bidID} \notin \boldsymbol{H}^{t-1}_{\mathsf{bid}}$, signatures are valid.

We consider the validity of bidpools instead of the correctness of UpdatePool because provers are not required to record old bidpools and bid sets from previous slots (as such requirements need enormous storage space). However, as we will show in the following section, the validity of bidpools is sufficient for defining the correctness of our PoBA scheme.

Algorithm 1: The UpdatePool algorithm. Let $t{\geq}1$ be the current slot. UpdatePool is parameterized by a bidpool Pool^{t-1}, a blockchain chain^{t-1}, and a set of bids P^{t-1}.

```
function UpdatePool($\mathsf{Pool}^{t-1}$, $\mathsf{chain}^{t-1}$, $P^{t-1}$);
Let $\mathsf{Pool}^t{=}\emptyset$;
if $t{=}1$ then
    Parse $\mathsf{Pool}^G{=}\emptyset$, $\mathsf{chain}{=}\mathsf{bk}^G$, and $P^G{=}\{(\mathsf{bidID{:}bid}){:}t_{\mathsf{Gen}}{=}G\}$;
    Return $\mathsf{Pool}^1{=}P^G$
else
    Parse the bid history of $\mathsf{chain}^{t-1}$ as $\boldsymbol{H}^{t-1}_{\mathsf{bid}}$ and the transaction history as $\boldsymbol{H}^{t-1}_{\mathsf{tx}}$;
    Parse $P^{t-1}{=}\{(\mathsf{bidID{:}bid}){:}t_{\mathsf{Gen}}{=}t{-}1\}$;
    // Add bids of slot $t{-}1$.
    Set $\mathsf{Pool}^t{=}\mathsf{Pool}^{t-1}{\cup}P^{t-1}$;
    // Add bids from $\boldsymbol{H}^{t-1}_{\mathsf{bid}}$.
    for $\mathsf{bid}{\in}\boldsymbol{H}^{t-1}_{bid}$ do
        if $\mathsf{bidID} \notin \mathsf{Pool}^{t-1}$ then
            Add (bidID:bid) to $\mathsf{Pool}^{t-1}$
        end
    end
    for $\mathsf{bidID}{\in}\mathsf{Pool}^t$ do
        Parse (bidID:bid)'s quantity and expiration slot as $q$ and $t_{\mathsf{Exp}}$;
        // Remove expired bids .
        if $t_{\mathsf{Exp}} < t$ then
            Remove (bidID:bid) from $\mathsf{Pool}^t$;
        end
        // Derive residuals given $\boldsymbol{H}^{t-1}_{\mathsf{tx}}$.
        Compute $q_{\mathsf{rbid}}{=}q - \sum_{\{\mathsf{tx}:\mathsf{txID}\in\boldsymbol{H}^{t-1}_{\mathsf{tx}}\wedge\mathsf{bidID}\in\mathsf{tx}\}} q_{\mathsf{tx}}$;
        if $q_{\mathsf{rbid}}{>}0$ then
            Replace with (bidID:rbid) of quantity $q_{\mathsf{rbid}}$;
        else
            Remove (bidID:bid) from $\mathsf{Pool}^t$;
        end
    end
    Return $\mathsf{Pool}^t$
end
```

B. Formal Syntax of PoBA

The PoBA scheme consists of a tuple of algorithms PoBA=(SampleBIDs, Solve, Eval). SampleBIDs samples an input bid set for the BAP; Solve outputs a block as the solution to the BAP, and it also outputs the extended blockchain; Eval first verifies the validity of the blockchain and outputs a score according to a public scoring function.

Definition 7 (PoBA Scheme): Let $\mathsf{Hash}{:}\{0,1\}^*{\to}\{0,1\}^\lambda$ be a collision-free hash function, and let $s_{\mathsf{bk}}{:}\mathbb{BK}{\to}\mathbb{R}$ be a publicly known scoring function as given in Definition 2. In slot $t{\geq}1$, for any prover $\mathcal{P}$, let $\mathsf{Pool}^t_{\mathcal{P}}$ be her updated bidpool from Algorithm 1, and let $\mathsf{chain}^{t-1}_{\mathcal{P}}{=}\mathsf{bk}^G||\mathsf{bk}^1||\ldots||\mathsf{bk}^{t-1}$ be her current blockchain. The prover performs (SampleBIDs, Solve), and any verifier performs Eval.

- $\mathsf{SampleBIDs}(\mathsf{Pool}^t_{\mathcal{P}}, \mathsf{N}; r^t_{\mathcal{P}})$ takes as input the prover's bidpool $\mathsf{Pool}^t_{\mathcal{P}}$, an upper bound N for the size of bid sets, and a random seed $r^t_{\mathcal{P}}$[3]. SampleBIDs outputs a set of bids $\mathsf{BIDs}^t_{\mathcal{P}} \subseteq \mathsf{Pool}^t_{\mathcal{P}}$ such that $|\mathsf{BIDs}^t_{\mathcal{P}}| \leq \mathsf{N}$;
- $\mathsf{Solve}(\mathsf{chain}^{t-1}_{\mathcal{P}}, \mathsf{BIDs}^t_{\mathcal{P}})$ takes as input the prover's blockchain and a bid set $\mathsf{BIDs}^t_{\mathcal{P}}$. It outputs a block candidate $\mathsf{bk}^t_{\mathcal{P}}{=}(\mathsf{prevHash}, \mathsf{BIDs}^t_{\mathcal{P}}, \mathsf{TXs}, \mathsf{aux})$ and $\mathsf{chain}^t_{\mathcal{P}}{=}\mathsf{chain}^{t-1}_{\mathcal{P}}||\mathsf{bk}^t_{\mathcal{P}}$. Here, $\mathsf{prevHash}{=}\mathsf{Hash}(\mathsf{bk}^{t-1})$;
- $\mathsf{Eval}(\mathsf{chain}^{t'}_{\mathcal{P}^*}, \mathsf{N})$ takes as input a blockchain $\mathsf{chain}^{t'}_{\mathcal{P}^*}$ from prover $\mathcal{P}^*$ and the size bound N for bid sets. Parse $\mathsf{chain}^{t'}_{\mathcal{P}^*}{=}\mathsf{chain}^{t'-1}||\mathsf{bk}^{t'}_{\mathcal{P}^*}$ where $\mathsf{bk}^{t'}_{\mathcal{P}^*}$ is generated by $\mathcal{P}^*$. It outputs $(1, s_{\mathsf{bk}}(\mathsf{bk}^{t'}_{\mathcal{P}^*}))$ if $\mathsf{chain}^{t'}_{\mathcal{P}^*}$ is valid; otherwise, it outputs $(0,\perp)$. Moreover, parse $\mathsf{bk}^{t\prime}_{\mathcal{P}^*}{=}(\mathsf{prevHash}, (\mathsf{BIDs}^*, \mathsf{TXs}^*, t'), \mathsf{aux}^*)$, we say $\mathsf{chain}^{t'}_{\mathcal{P}^*}$ is valid if the following conditions hold:

1) $t'{=}t$, *i.e.*, the given blockchain is in the current slot. Then, we use t instead of t' in the following conditions;
2) $(1,\cdot){\leftarrow}\mathsf{Eval}(\mathsf{chain}^{t-1}, \mathsf{N})$, *i.e.*, the blockchain get extended is valid;
3) The previous hash satisfies $\mathsf{prevHash}{=}\mathsf{Hash}(\mathsf{bk}^{t-1})$;
4) The bid set satisfies that: (1) $|\mathsf{BIDs}^*|{\leq}\mathsf{N}$; (2) for each $\mathsf{bid}{\in}\mathsf{BIDs}^*$ with generation and expiration slots $(t_{\mathsf{Gen}}, t_{\mathsf{Exp}})$, the current slot $t{\in}[t_{\mathsf{Gen}}, t_{\mathsf{Exp}}]$; (3) let bidID be $\mathsf{bid}{\in}\mathsf{BIDs}^*$'s identifier, for any $\mathsf{bidID} \notin \boldsymbol{H}^{t-1}_{\mathsf{bid}}$, the signatures in bid are valid;
5) Let $BI{=}\{\mathsf{bidID{:}bid}{\in}\mathsf{BIDs}^*\}$ and $TI{=}\{\mathsf{txID{:}tx}{\in}\mathsf{TXs}^*\}$ be the identifier sets given by BIDs^* and TXs^*, respectively. The transaction set satisfies that: (1) for each $\mathsf{tx}{=}(\mathsf{bid}_1, \mathsf{bid}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}}){\in}\mathsf{TXs}^*$, $\mathsf{bidID}_1, \mathsf{bidID}_2{\in}\boldsymbol{H}^{t-1}_{\mathsf{bid}}{\cup}BI$; (2) Let $\boldsymbol{H}^t_{\mathsf{tx},\mathcal{P}^*}{=}\boldsymbol{H}^{t-1}_{\mathsf{tx}}{\cup}TI$. For each $\mathsf{tx}{=}(\mathsf{bid}_1, \mathsf{bid}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}}){\in}\boldsymbol{H}^t_{\mathsf{tx},\mathcal{P}^*}$, let $\mathsf{bid}_1, \mathsf{bid}_2$ be a buy bid and a sell bids with identifiers $\mathsf{bidID}_1, \mathsf{bidID}_2$, quantity q_1, q_2 and price p_1, p_2, respectively. The assigned quantity and price of tx satisfies $\sum_{\mathsf{txID}\in\boldsymbol{H}^t_{\mathsf{tx},\mathcal{P}^*}\wedge\mathsf{bid}_1\in\mathsf{tx}} q_{\mathsf{tx}} \leq q_1 \bigwedge \sum_{\mathsf{txID}\in\boldsymbol{H}^t_{\mathsf{tx},\mathcal{P}^*}\wedge\mathsf{bid}_2\in\mathsf{tx}} q_{\mathsf{tx}} \leq q_2 \bigwedge p_2 \leq p_{\mathsf{tx}} \leq p_1$ (which is equivalent to Equation 1).
6) The auxiliary aux^* satisfies that: (1) the block's generation slot $t^*{=}t$; (2) signatures are valid.

[3]We write the randomness explicitly for later modeling PoBA in Section V-C

The correctness of PoBA is defined as follows.

Definition 8 (Correctness of PoBA): For any prover $\mathcal{P}$ in slot $t{\geq}1$, let $(\mathsf{Pool}^t, \mathsf{chain}^{t-1}, \mathsf{N})$ be the prover's input tuple such that Pool^t is valid with respect to chain^{t-1} as defined in Definition 6. The PoBA scheme is correct, if $\mathsf{BIDs} \leftarrow \mathsf{SampleBIDs}(\mathsf{Pool}^t, \mathsf{N})$ and $\mathsf{chain}^{t-1}_{\mathcal{P}}||\mathsf{bk}^t_{\mathcal{P}} \leftarrow \mathsf{Solve}(\mathsf{chain}^{t-1}_{\mathcal{P}}, \mathsf{BIDs})$ are honestly executed, then $\Pr\left[(1, s_{\mathsf{bk}}(\mathsf{bk}^t_{\mathcal{P}})) \leftarrow \mathsf{Eval}(\mathsf{chain}^{t-1}_{\mathcal{P}}||\mathsf{bk}^t_{\mathcal{P}}, \mathsf{N})\right] = 1$.

In PoX schemes that involve computational tasks [11], [12], [16] difficulty is crucial. Intuitively, it requires provers to contribute enough computing power to generate valid blocks. Otherwise, they can generate massive amounts of blocks in a short time period so that the network fails to finalize on a chain of blocks. However, as mentioned in Section III-B, finding a valid BAP solution does not require much computing power. Hence, it is easy to generate valid blocks for any prover in our PoBA scheme. To tackle this difference, next we propose a "highest-score" rule for chain selection.

V. Design and Security of Our Protocol

Each user can generate and diffuse multiple valid blocks, then we utilize a directed tree (precisely, forest due to potentially missing blocks) to store blocks locally. The highest-score rule requires honest users to extend their block-tree with newly received blocks and select the highest-scored branch on the tree as their blockchain. Hence, we further define the score of branches to support this selection mechanism.

A. Block-Tree and Branch Score

Starting with definitions, we first consider a master-tree of slot $t{\geq}1$, denoted by mtree^t, that contains all valid blocks generated (*not* diffused) by users from the genesis slot 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, 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, honest users may only see a part of the master-tree. Hence, a user's view, denoted by $\mathsf{tree}^t_{\mathcal{U}}$, is a sub-tree of the master tree.

Definition 9 (Master-Tree and User-Tree): Let bk^G be the genesis block. For any $\ell{\geq}1$ and all $i{\in}[n]$ where n is the number of users participating the protocol in slot $t{\in}[\ell]$, let $BK^t_i{=}\{\mathsf{bk}^t_i\}{\neq}\emptyset$ denote the set of valid blocks generated by user $\mathcal{U}_i$. Then, $BK^t{=}\bigcup_{i\in[n]} BK^t_i$ denotes the set of all valid blocks generated in slot t. The master-tree of slot ℓ is defined as $\mathsf{mtree}^\ell{=}(V, E)$ where $V{=}\{\mathsf{bk}^G\}{\cup}\bigcup_{t\in[\ell]} BK^t$, and $E{=}\{(\mathsf{bk}^G, \mathsf{bk}^1){:}\forall \mathsf{bk}^1{\in}BK^1\}{\cup}\bigcup_{t\in[t-1]}\{(\mathsf{bk}^t, \mathsf{bk}^{t+1}){:}\forall \mathsf{bk}^t{\in} BK^t, \mathsf{bk}^{t+1}{\in}BK^{t+1}, \mathsf{prevHash}{=}\mathsf{Hash}(\mathsf{bk}^t)\}$. Here, $\mathsf{prevHash}$ is the previous hash value in block bk^{t+1}, *i.e.*, blocks $(\mathsf{bk}^t, \mathsf{bk}^{t+1})$ are linked by the hash function in PoBA (Definition 7). A user-tree of $\mathcal{U}$, denoted by $\mathsf{tree}^\ell_{\mathcal{U}}{=}(V_{\mathcal{U}}, E_{\mathcal{U}})$, satisfies $\mathsf{tree}^\ell_{\mathcal{U}}{\subseteq}\mathsf{mtree}^\ell$.

Next, we define branches with respect to a given block-tree (master or user), which is a chain of blocks (Definition 2).

Definition 10 (Branch on Tree): Let $G^\ell{=}(V^\ell, E^\ell)$ be a block-tree (master or user) of slot $\ell{\geq}1$. A branch is defined as $\mathsf{branch}^t{=}\mathsf{bk}^G||\mathsf{bk}^1_{i_1}||\ldots||\mathsf{bk}^t_{i_t}$ such that $\mathsf{branch}^t \subseteq G^\ell$ and the vertex represents $\mathsf{bk}^t_{i_t}$ on G^ℓ is a leaf vertex.

Later, we may distinguish the notations of branch and chain by using branch^ℓ for an arbitrary branch on a given block-tree, and chain^ℓ for the highest-scored branch.

To define the score of branches, we introduce an accumulating function $acc : \mathbb{N} \to \mathbb{R}$ such that it takes as input the height of vertices on the given branch.

Definition 11 (Score of Branches): Let $\mathsf{branch}^t \subseteq G^\ell$ be a branch as in Definition 10 where $\ell{\geq}1$ and $t \in [\ell]$. Parse it with $\mathsf{branch}^t{=}\mathsf{bk}^G||\mathsf{bk}^1_{i_1}||\ldots||\mathsf{bk}^t_{i_t}$. The score of branch^t is defined as follows

$$S_{\mathsf{branch}^t}{=}\sum_{i=1}^{t} acc(t){\cdot}s_{\mathsf{bk}}(\mathsf{bk}^i). \tag{4}$$

Therefore, our highest-score rule requires honest users with $\mathsf{tree}^\ell_{\mathcal{U}}$ to adopt chain^ℓ such that $S_{\mathsf{chain}^\ell}{=}\max_{\mathsf{branch}^t\subseteq\mathsf{tree}^\ell_{\mathcal{U}}} S_{\mathsf{branch}^t}$ as their selected blockchain.

B. Protocol Description

This section presents the workflow of an honest user $\mathcal{U}$ in an arbitrary slot $t{\geq}1$. Denote the user's view of bidpool and block-tree at the end of slot $t{-}1$ with Pool^{t-1} and tree^{t-1}, respectively. Here, we do not specify them to $\mathcal{U}$ because the permissionless setting cannot guarantee the user to participate in slot $t{-}1$. However, we require that any branch $\mathsf{branch}{\subseteq}\mathsf{tree}^{t-1}$ satisfies $(1, \cdot){\leftarrow}\mathsf{PoBA.Eval}(\mathsf{branch}, \mathsf{N})$.

Let $\mathsf{chain}^{t-1}_{\mathcal{U}}$ be the highest-scored branch in tree^{t-1}, and let $P^{t-1}{=}\{\mathsf{bid}{:}t_{\mathsf{Gen}}{=}t{-}1\}$ be the set of bids that are issued in slot $t{-}1$. The user first executes the $\mathsf{UpdatePool}$ algorithm from $(\mathsf{Pool}^{t-1}, \mathsf{chain}^{t-1}_{\mathcal{U}}, P^{t-1})$ to obtain the updated bidpool $\mathsf{Pool}^t_{\mathcal{U}}$. Then, she performs the PoBA scheme to obtain block candidates (the output will be a set). Denote the highest-scored candidate with $\mathsf{bk}^t_{\mathcal{U}}$. The user diffuses her extended blockchain with $\mathsf{chain}^t_{\mathcal{U}}{=}\mathsf{chain}^{t-1}_{\mathcal{U}}||\mathsf{bk}^t_{\mathcal{U}}$. Moreover, she receives blockchains from others. The process of honest users updating their local block-tree with each incoming blockchain is specified by an $\mathsf{UpdateTree}$ algorithm in Algorithm 2.

Finally, the user is responsible for reporting her confirmed blockchain, *i.e.*, $\mathsf{chain}^t_{\mathcal{U}} \subseteq \mathsf{tree}^t_{\mathcal{U}}$, to the protocol with respect to a parameter k (to be estimated in Section V-C).

C. Security Analysis

To analyze the security of our protocol, we first model the general scoring function-based PoBA scheme with a universal sampler [9]. The formal definition can be found in [9], [10]. We omit it due to page limitations.

Recall that hash computation in proof-of-work can modeled with queries to random oracle [2]. Following the same approach, we enable each honest user to make at most $q > 0$ queries to the universal sampler; whereas, the adversary can make at most $q_{\mathcal{A}} > q$ queries. This difference in query capabilities indicates the difference in computing power between honest and adversarial provers. However, the total number of queries in each slot is bounded above by $Q \in \mathbb{N}$. We further clarify that the network or the adversary cannot delay queries

Algorithm 2: The $\mathsf{UpdateTree}$ algorithm. Let $t{\geq}1$ be the current slot. $\mathsf{UpdateTree}$ is parameterized by a block-tree tree and a blockchain candidate $\mathsf{chain}^{t'}_{\mathcal{U}^*}$.

```
1  function UpdateTree($\mathsf{tree}, \mathsf{chain}^{t'}_{\mathcal{U}^*}$);
2  Parse $\mathsf{tree}{=}\{V, E\}$;
   // Verify the incoming blockchain.
3  Run $(b, \cdot) \leftarrow \mathsf{PoBA.Eval}(\mathsf{chain}^{t'}_{\mathcal{U}^*}, \mathsf{N})$;
4  if $b{=}1$ then
       // $t'{=}t$.
5      Parse $\mathsf{chain}^{t'}_{\mathcal{U}^*}{=}\mathsf{bk}^G||\mathsf{bk}^1_{\mathcal{U}^*}||\ldots||\mathsf{bk}^{t-1}_{\mathcal{U}^*}||\mathsf{bk}^t_{\mathcal{U}^*}$;
       // Find the first block not in tree.
6      for $\mathsf{bk}^t_{\mathcal{U}^*} \in \mathsf{chain}^{t'}_{\mathcal{U}^*}$ do
7          if $\mathsf{bk}^{t-1}_{\mathcal{U}^*} \in \mathsf{tree}$ and $\mathsf{bk}^t_{\mathcal{U}^*} \notin \mathsf{tree}$ then
8              Set $V'{=}V \cup \{\mathsf{bk}^t_{\mathcal{U}^*}, \ldots, \mathsf{bk}^t_{\mathcal{U}^*}\}$;
9              Set $E'{=}E \cup$
                 $\{(\mathsf{bk}^{t-1}_{\mathcal{U}^*}, \mathsf{bk}^t_{\mathcal{U}^*}), \ldots (\mathsf{bk}^{t-1}_{\mathcal{U}^*}, \mathsf{bk}^t_{\mathcal{U}^*})\}$;
10             Return $\mathsf{tree}^t{=}(V', E')$
11         end
12     end
13 end
```

(the communication between users and the universal sampler), given it is local oracle access. The purpose of the universal sampler is to model the output distribution of PoBA relying only on users' randomness and solving algorithms.

Therefore, we require the universal sampler to have the single property of randomly sampling a block $\mathsf{bk}{\in}\mathbb{BK}$ such that $s_{\mathsf{bk}}(\mathsf{bk}) \overset{\mathcal{D}}{\leftarrow} \mathcal{S}$ where $\mathcal{D}$ and $\mathcal{S}$ denotes the score distribution and score space determined by the general scoring function. We also require that blocks are strictly ordered by the scoring function with high probability (which can be achieved by considering generation time). Finally, we specify the output distribution of the universal sampler, which is modeled with a continuous random variable X following distribution $\mathcal{D}$ over a score space $\mathcal{S}{=}[\mathsf{smin}, \mathsf{smax}]$. Here, $\mathcal{D}, \mathsf{smin}, \mathsf{smax}$ are determined by the general scoring function $s_{\mathsf{bk}}(\cdot)$. We denote the probability density function and the distribution function of $\mathcal{D}$ with $f(\cdot)$ and $F(\cdot)$ such that $F(x){=}\Pr[X{\leq}x]{=}\int_{\mathsf{smin}}^{x} f(t)dt$.

Next, we consider persistence and liveness [2] as core security properties for our protocol. For persistence, we adopt a slightly refined version from [12]. Whereas for liveness, as proven in [2], it is derived from two basic properties called chain growth and chain quality. We observe that: (1) chain growth follows directly from our slot-based execution setting as explained in Section II-A; (2) our score-based design can eliminate the necessity of chain quality. We explain the reason of our second observation as follows. Chain quality requires the fraction of adversarial blocks within 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, adversaries cannot gain advantages from such an attack because it will lower the score of blocks, hence, lowering the possibility of the adversarial blocks being selected. Based on these observations, we define a "block-liveness" property for our protocol instead of the conventional liveness for transactions.

Definition 12 (Persistence and Block-Liveness): Formal definitions are as follows.

- Persistence: For any two honest users with blockchains $\mathsf{chain}^{t_1}_1, \mathsf{chain}^{t_2}_2$ at slot $t_1, t_2{\geq}1$, respectively. Without loss of generality, let $t_1{\leq}t_2$. Persistence with parameter $k \in \mathbb{N}$ indicates that $\mathsf{chain}^{t_1}_1$, is a prefix of $\mathsf{chain}^{t_2}_2$ after removing the rightmost k blocks;
- Block-liveness: For any honest user with chain^t in slot $t{\geq}1$, block-liveness states that for any $i \in [t]$, chain^i is extended by at exact one block.

Consider an honest user's local block-tree, then persistence is proven by: (1) The highest-scored branch of each slot will eventually be known to all honest users; (2) Selected branches of different slots should have a long enough common prefix.

If a block or branch is known to all honest users, it is disclosed. Then, the δ-bounded network guarantees that block candidates generated by honest users are always disclosed after δ slots. For block/branch disclosure, thereby the next lemma.

Lemma 1: Given a δ-synchronous network, for any slot $t{\geq}1$, if $\mathsf{branch}^t \subseteq \mathsf{mtree}^t$ is the highest-scored branch in t as in Definition 11, branch^t will be disclosed for any $\ell{\geq}t + \delta + 1$.

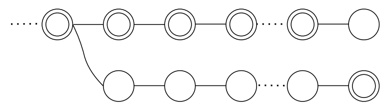
Recall that we empower it to learn all generated blocks. Hence, the adversary knows the master-tree in any given slot. We further require the adversary to send the highest-scored blockchain of each slot to at least one honest user.

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

Then, the proof of Lemma 1 goes as follows.

Proof: Consider the rightmost block on $\mathsf{branch}^t{=}\mathsf{bk}^G||\ldots||\mathsf{bk}^t$. 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 branch^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 branch^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 $\ell{\geq}t + \delta + 1$, branch^t is disclosed.

However, knowing the highest-scored branches of each time 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. We consider two persistence violations (In the illustration, the circle denotes the blocks on the branches, and the double circle denotes the branch being selected as the highest-scored one): (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 happens frequently so that consensus cannot be settled (Figure 1b).


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

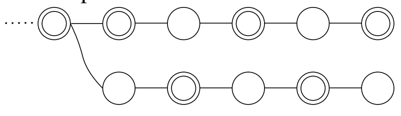

(b) Chains **swing** over conjunctive slots.

Fig. 1: Persistence Violations: Substitution and Swing.

We consider both violation cases with parameter $\tau{\geq}1$. For the first case, we have the following lemma, which analyzes the probability of any branch that deviates the selected chain for τ slots getting selected in slot t. Here, we consider the selected chain satisfies that $\mathsf{chain}^{t-\tau}$ gets selected conjunctively for another $\tau{\geq}1$ slots, *i.e.*, for any $i \in [t-\tau, t-1]$, there exists bk^i such that $\mathsf{chain}^i{=}\mathsf{chain}^{i-1}||\mathsf{bk}^i$.

Lemma 2: Let $\mathcal{D}$ be the range distribution of any arbitrary scoring function $s_{\mathsf{bk}}(\cdot)$ used in PoBA. Let mtree^t be the master-tree of any slot $t{>}\tau$. For any slot $i \in [t-\tau, t-1]$, let chain^i be the highest-scored branch such that $\mathsf{chain}^i{=}\mathsf{chain}^{i-1}||\mathsf{bk}^i$. Assuming at least one honest user, there exists an accumulating function $acc(\cdot)$ (Definition 11) such that the probability of any branch that satisfies $\mathsf{branch}^t \cap \mathsf{chain}^{t-1}{=}\mathsf{chain}^{t-\tau-1}$ getting selected in slot t is $\mathsf{O}(c^{-\tau})$. Here, $c>1$ is a constant value given by $acc(\cdot)$.

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

$$\Pr[s(\mathsf{bk}_\mathsf{c}^{t-1}) - s(\mathsf{bk}_\mathsf{b}^{t-1}){\geq}0{\wedge}c_{0,1}{\cdot}\left(s(\mathsf{bk}_\mathsf{c}^{t-1}) - s(\mathsf{bk}_\mathsf{b}^{t-1})\right) + \left(s(\mathsf{bk}_\mathsf{c}^{t}) - s(\mathsf{bk}_\mathsf{b}^{t})\right) {\leq}0]. \tag{5}$$

Consider an accumulating function such that $\frac{acc(t-1)}{acc(t)}{=}c_{0,1}$. Denote random variables that represents the scores of the tuple $(\mathsf{bk}_\mathsf{c}^{t-1}, \mathsf{bk}_\mathsf{b}^{t-1}, \mathsf{bk}_\mathsf{c}^{t}, \mathsf{bk}_\mathsf{b}^{t})$ with $(X_\mathsf{c}^{t-1}, X_\mathsf{b}^{t-1}, X_\mathsf{c}^{t}, X_\mathsf{b}^{t})$. Then, write the subtraction of scores with $Y^{t-1}{=}X_\mathsf{c}^{t-1} - X_\mathsf{b}^{t-1}$ and $Y^t{=}X_\mathsf{c}^t - X_\mathsf{b}^t$. Following the universal sampler model, $(X_\mathsf{c}^{t-1}, X_\mathsf{b}^{t-1}, X_\mathsf{c}^{t}, X_\mathsf{b}^{t})$ are independent and follow the same distribution $\mathcal{D}_X{=}\mathcal{D}$ on $[\mathsf{smin}, \mathsf{smax}]$. Hence, Y^{t-1} and Y^t are independent, and distributed identically and *symmetrically* [17] on $[\mathsf{smin}{-}\mathsf{smax}, \mathsf{smax}{-}\mathsf{smin}]$. We denote the distribution of Y^{t-1} and Y^t 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

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

Consider the event: $\{Y^{t-1}{=}y{\wedge}Y^t{\leq}{-}c_{0,1}y\}$ for all $y{\in}[0, \mathsf{smax}{-}\mathsf{smin}]$. For simplicity, we rewrite $r{=}\mathsf{smax}{-}\mathsf{smin}$. By Y^{t-1} being independent of Y^t, we have Equation $6{=}\int_0^r \int_{-r}^{-c_{0,1}y} f_Y(y) f_Y(x) \mathrm{d}x\mathrm{d}y$. 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 Equation $6{\leq} \int_0^{\infty} \int_{-\infty}^{-c_{0,1}y} e^{-y^2}{\cdot}e^{-x^2}\mathrm{d}x\mathrm{d}y{=}\frac{{c_1}^2}{2c_2}{\cdot}\tan^{-1}\left(\frac{1}{c_{0,1}}\right)$ ${\leq}\frac{{c_1}^2}{2c_2\cdot c_{0,1}}$. That is, $\Pr[\tau{=}1, \mathsf{branch}_\mathsf{b}]{\leq}{c_1}^2/(2c_2{\cdot}c_{0,1})$ for any $c_1, c_2{\geq}0$ and $c_1^2/c_2{\geq}\pi^{-1}$ where $c_{0,1}{=}\frac{acc(t-1)}{acc(t)}$.

Next, the probability for any ${\mathsf{branch}_\mathsf{b}}^t$ ($\Pr[\tau{=}1]$) should consider a joint event, *i.e.*, let q be the number of possible $\mathsf{branch}_\mathsf{b}$ in slot t. 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 $\Pr[\tau{=}1]{\leq}1-\left(1-\frac{c_1^2}{2c_2\cdot c_{0,1}}\right)^Q$. Thus $\tau>1$ follows the same methodology of $\tau{=}1$. Therefore, we consider a constant value $c > \max\{1, \frac{c_1^2}{c_2}\}$, and let $acc(t){=}c^t$. Then, $\Pr[\tau{\geq}1]{\leq}1-\left(1-\frac{c_1^2}{c_2\cdot c^{-\tau}}\right)^Q$ 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 for at least one honest user. Finally, we can conclude that the first violation case occurs with probability $\mathsf{O}(c^{-\tau})$.

Now, we analyze the second violation case. It occurs only when the adversary finds the highest-scored *block* for multiple conjunctive slots. This is because honest users will stick to the highest-scored branch and not work on an inferior branch.

Lemma 3: Let $\mathcal{D}$ be the range distribution of any arbitrary scoring function $s_{\mathsf{bk}}(\cdot)$ used in PoBA. Let mtree^t be the master-tree of any slot $t{>}\tau$. Assuming the fraction of adversarial computing power is f, the probability of chain swinging during $\tau{\geq}1$ conjunctive slots is $\mathsf{O}(f^{-\tau})$.

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, 2, 3, thereby the theorem for persistence.

Theorem 1 (Persistence): Assuming the fraction of adversarial computing power is f, the protocol parameterized by $k{\geq}\delta+1$ satisfies persistence with probability at least $1-\Omega(\max\{c, f^{-1}\}^{-k+\delta})$.

For completeness, we show the following theorem for block-liveness. 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.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, http://bitcoin.org/bitcoin.pdf.

[2] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, ser. Lecture Notes in Computer Science, E. Oswald and M. Fischlin, Eds., vol. 9057. Springer, 2015, pp. 281–310. [Online]. Available: https://doi.org/10.1007/978-3-662-46803-6_10

[3] T. Górski and J. Bednarski, "Modeling of smart contracts in blockchain solution for renewable energy grid," in *Computer Aided Systems Theory - EUROCAST 2019 - 17th International Conference, Las Palmas de Gran Canaria, Spain, February 17-22, 2019, Revised Selected Papers, Part I*, ser. Lecture Notes in Computer Science, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., vol. 12013. Springer, 2019, pp. 507–514. [Online]. Available: https://doi.org/10.1007/978-3-030-45093-9_61

[4] R. Akhras, W. El-Hajj, M. Majdalani, H. M. Hajj, R. A. Jabr, and K. B. Shaban, "Securing smart grid communication using ethereum smart contracts," in *16th International Wireless Communications and Mobile Computing Conference, IWCMC 2020, Limassol, Cyprus, June 15-19, 2020*. IEEE, 2020, pp. 1672–1678. [Online]. Available: https://doi.org/10.1109/IWCMC48107.2020.9148345

[5] D. Kirli, B. Couraud, V. Robu, M. Salgado-Bravo, S. Norbu, M. Andoni, I. Antonopoulos, M. Negrete-Pincetic, D. Flynn, and A. Kiprakis, "Smart contracts in energy systems: A systematic review of fundamental approaches and implementations," *Renewable and Sustainable Energy Reviews*, vol. 158, p. 112013, 2022.

[6] L. Luu, D. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 254–269. [Online]. Available: https://doi.org/10.1145/2976749.2978309

[7] H. Robertson, "Ethereum transaction fees are running sky-high." 2021, "https://markets.businessinsider.com/news/currencies/ethereum-transaction-gas-fees-high-solana-avalanche-cardano-crypto-blockchain-2021-12".

[8] J. S. Park, B. H. Lim, and Y. Lee, "A lagrangian dual-based branch-and-bound algorithm for the generalized multi-assignment problem," *Manage. Sci.*, vol. 44, no. 12, p. 271–275, dec 1998.

[9] D. Hofheinz, T. Jager, D. Khurana, A. Sahai, B. Waters, and M. Zhandry, "How to generate and use universal samplers," in *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, J. H. Cheon and T. Takagi, Eds., vol. 10032, 2016, pp. 715–744. [Online]. Available: https://doi.org/10.1007/978-3-662-53890-6_24

[10] J. Blocki and H. Zhou, "Designing proof of human-work puzzles for cryptocurrency and beyond," in *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, M. Hirt and A. D. Smith, Eds., vol. 9986, 2016, pp. 517–546. [Online]. Available: https://doi.org/10.1007/978-3-662-53644-5_20

[11] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of work from worst-case assumptions," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, H. Shacham and A. Boldyreva, Eds., vol. 10991. Springer, 2018, pp. 789–819. [Online]. Available: https://doi.org/10.1007/978-3-319-96884-1_26

[12] M. Fitzi, A. Kiayias, G. Panagiotakos, and A. Russell, "Ofelimos: Combinatorial optimization via proof-of-useful-work \\ A provably secure blockchain protocol," in *Advances in Cryptology - CRYPTO 2022*, ser. Lecture Notes in Computer Science. Springer, 2022.

[13] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988. [Online]. Available: https://doi.org/10.1137/0217017

[14] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145. [Online]. Available: https://doi.org/10.1109/SFCS.2001.959888

[15] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," in *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, ser. Lecture Notes in Computer Science, J. B. Nielsen and V. Rijmen, Eds., vol. 10821. Springer, 2018, pp. 3–33. [Online]. Available: https://doi.org/10.1007/978-3-319-78375-8_1

[16] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, ser. Lecture Notes in Computer Science, E. F. Brickell, Ed., vol. 740. Springer, 1992, pp. 139–147. [Online]. Available: https://doi.org/10.1007/3-540-48071-4_10

[17] G. C. T. George E.P. Box, *Bayesian Assessment of Assumptions 1. Effect of Non-Normality on Inferences about a Population Mean with Generalizations*. John Wiley and Sons, Ltd, 1992, ch. 3, pp. 149–202. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118033197.ch3