Peer-to-Peer Energy Trading Meets Blockchain: Consensus via Score-Based Bid Assignment

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

[1] School of Computing, Tokyo Institute of Technology. Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55. su.x.ab@m.titech.ac.jp, mario@c.titech.ac.jp, keisuke@is.titech.ac.jp.

[2] Input Output, Global. mario.larangeira@iohk.io.

[3] Mitsubishi Electric.

Abstract. The demand for peer-to-peer energy trading (P2PET) grows alongside the advancement of smart grids. A P2PET system enables its peers to trade energy as in a double-sided auction market by issuing auction bids to buy or sell energy. A robust public ledger, that satisfies the standard properties of persistence and liveness, is necessary for the system to record trading agreements, i.e., combinations between buy and sell bids which would form a *transaction*. The Bitcoin based blockchain satisfies such properties as proven in the backbone protocol (EuroCrypt'15). However, existing blockchain-based P2PET approaches rely on general-purpose blockchains with smart contract capabilities, unavoidably incurring in high operational costs. Therefore, this work intends to design a dedicated blockchain for the ledger of P2PET. We first revisit the blockchain data structure to support auction bids. Then, we abstract the process of forming transactions, i.e., matching bids, with a score-based many-to-many Bid Assignment Problem (BAP). Leveraging our proposed BAP in addition to the corresponding scoring function, we propose a "proof-of" scheme, namely Proof-of-Bid-Assignment (PoBA), and design the corresponding blockchain aided protocol. The key difference from any previous work, we are aware of, is that our protocol selects blocks according to the score of their content, i.e., bids and transactions. Hence, a higher-scored block would be preferable to the underlying P2PET system regardless of whether the block's generator is honest, since, intuitively, it would increase the number of trading agreements. Finally, by modeling PoBA with a universal sampler (AsiaCrypt'16) and analyzing honest users' local chain dynamics, we prove the security of our design with respect to the standard ledger properties.

Keywords: Smart Grid, Peer-to-Peer Energy Trading System, Generalized Multi-Assignment Problem, Score-Based Blockchain Consensus.

1 Introduction

We start by reviewing the current P2PET scenario and its relation with blockchain-aided protocols before describing our approach and contribution.

1.1 Background and Motivation

The shift to renewable but less reliable energy sources urges a significant change in the current grid structure, from the centralized uni-direction traditional system to the decentralized bi-direction “smart grid”. A smart grid involves prosumers who both produce and consume energy. The infrastructure enables its users, *i.e.*, the early prosumers, to trade energy and exchange data on demand, hence contributing to regional self-sustainability. A control system, *i.e.*, the P2PET system, monitors and manages the user operations within the smart grid. The paramount functionality is to securely record the trading history so that users can perform accordingly. In a decentralized environment, a public ledger that ensures user consensus is usually used for this purpose, to prevent any single party from tampering with its data.

The problem of consensus, where multiple participants are made to agree on common decisions, has been studied for at least five decades in the context of distributed systems. Due to the popularity of Bitcoin [20] 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 [12]. It is hence natural to utilize a blockchain as the implementation of a distributed ledger in P2PET systems.

Related works. Numerous studies investigate the use of blockchains in P2PET [1, 15,18], however these studies are built atop general-purpose blockchain protocols and rely on smart contract capabilities [19], *e.g.*, as provided in Ethereum [24] and Cardano [10]. Unfortunately, this severely limits the opportunities for optimizations, and these protocols suffer from high maintenance fees [23], where users must incur severe costs while submitting smart contract-based transactions to the network. Often the mining mechanism is typically based on proof-of-work (PoW) (*e.g.*, enforcing the repeated computation of huge amounts of hash function evaluations) which requires significant computational power (and hence energy), a sharp contrast with the goals of improving *energy trading* and reducing energy consumption. Needless to say, such repeated evaluations, as a computational problem, have no connection with the market that may exist on the top of the system. This gap is the starting point of our work.

This work explores the design of a dedicated blockchain protocol for P2PET, purposely deviating from the use of smart contracts and PoW. We thoroughly redesign, from the very bottom, the blockchain data structure itself. Hence, we can exploit the operations in the P2PET system and integrate them into a novel “proof-of-X” approach, named Proof-of-Bid-Assignment (PoBA) (to be detailed later). Therefore, unlike the PoW paradigm, in which much computing (and electrical) power is wasted in order to pace the block generation as a cornerstone for ensuring a robust ledger, ours leverages the existing architecture and structure of power distribution to provide a leaner and more effective solution. In summary, our novel framework integrates the underlying network mechanism, to ensure the robust ledger, with the market dynamics of the system.

1.2 Our Approach and Contributions

Now, we show a brief image of our approach. The first step is to extract settings and unique operations from the P2PET system.

Abstracted P2PET. A P2PET apparatus is the control system of the underlying smart grid, executed among users and potentially regional power plants within a *tight-knit* community, *e.g.*, a town. The smart grid infrastructure provides users with equipment to produce, store, and transmit energy. The users are also equipped with certificated, hence, tamper-proof smart-meters that measure energy production and consumption. Each smart-meter is associated with a Home Energy Management System (HEMS) that oversees the process of energy management. Unlike smart-meters, 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 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 considers a typical standalone P2PET system of a small community in which users and a power plant are connected via bi-direction power lines given a fixed physical topology. Each user can produce or consume energy freely. However, due to the unreliability of the energy production, the user may face (1) shortages when not producing enough, or (2) excesses when producing too much. Here, the trading capability of the P2PET system becomes crucial since it enables the user to buy or sell energy. Note that the smart grid infrastructure also enables users to store energy. Hence, in addition to buying in shortage and selling in excess, users may also buy for storing when the energy price is low and sell for profit when the price is high. We do not specify users' purpose but focus on the buy/sell operation. In order to support these operations, recall a double-sided auction market in which users can buy some product at a price by issuing a buy bid, and sell some product at a price by issuing a sell bid. We follow the same process of bidding with energy as the product.

Next, we consider the formation of trading agreements from the bids. Each agreement (later called transaction) consists of a buy bid and a sell bid that satisfies some constrain, *e.g.*, the buy bid's price should be higher than the sell bid's price. The public ledger of P2PET should record properly formed transactions, thus users can transmit energy and payment accordingly. It may have been noticed that two crucial questions are not mentioned above: (1) Given a set of bids, how to generate a set of transactions; (2) How to ensure a robust ledger in such a setting. The following sections thoroughly answer these questions. Moreover, we need to emphasize that this work focuses on the design of the blockchain protocol level instead of the P2PET system or smart grids. Therefore, we will *NOT* fully investigate the dynamics of the energy trading market or problems in the physical infrastructure, *e.g.*, energy transmission loss. However, we argue that our protocol is general enough to cover such problems.

P2PET meets blockchain: A brief description. In order to integrate the P2PET bidding operations with blockchain protocols, we first refine the blockchain data structure. Concretely, we add the bid layer in which each bid contains a type, *i.e.*, buy or sell, a quantity, and a unit price. In our setting, the blockchain transaction is the combination of a buy and a sell bid. Briefly, the block is defined as a container of bids and transactions, and the blockchain is defined as an ordered linked list of blocks.

Next, we formalize the process of generating transactions with a many-to-many assignment problem, *i.e.*, the Bid Assignment Problem (BAP), which can be further regarded as a special case of the Generalized Multiple-Assignment Problem (GMAP) [21]. We start by defining a generic framework of the BAP in which a set of bids is taken as input, and an index set of assigned bid pairs is output according to a scoring function. Then, by instantiating the BAP's outputs with blocks and specifying the scoring function's domain on the block's space, we instantiate the BAP to put forth the *BAP for block generation* (bk-BAP). Finally, we leverage the bk-BAP as the base problem of our "proof-of" scheme, the Proof-of-Bid-Assignment (PoBA) scheme.

Before proposing the formal syntax of the PoBA scheme, we introduce the concept of bidpool, *i.e.*, a pool of bids, which mimics the mempool of transactions in conventional blockchain protocols. We enable each PoBA user (formally called prover) to maintain and verify a bidpool according to her view of the blockchain. Then, we define the PoBA scheme that consists of algorithms for (1) sampling a bid set, from the bidpool, (2) solving the bk-BAP, given by the sampled bid set, and (3) evaluating the bk-BAP solution, according to the public scoring function. Here, the scoring function can enforce particular market dynamics, *e.g.*, by granting different scores to particular transactions. Since we will not fully investigate the underlying market and want to showcase the flexibility of our design, we leave the scoring function as general as possible. Then, when analyzing the computation in PoBA with respect to the general scoring function, we adopt the universal sampler [5,16] that samples blocks and the corresponding scores following arbitrary distributions. Our modeling follows a similar approach as PoW-based blockchain protocols, in which the hash computation in PoW is modeled by the query to a random oracle [12].

The earlier outlined bid-related block design is one of the major technical novelties of our proposal. To the best of our knowledge, this work is the first to introduce such an approach. Moreover, another unique feature of our protocol lies in the block selection and blockchain dynamics. Note that we do not require *optimality* in each user's block generation, *i.e.*, briefly, the best possible bid match combination, but instead, we encourage them to compete with each other so that the protocol can eventually put forth a chain of blocks with the highest overall score. Hence, we consider a tree structure for maintaining blocks that users see in the protocol, *i.e.*, the local view. Each branch on the block-tree is a valid chain of blocks. Then, we assign branches with scores based on the score of each block in the branch, thereby users can select the highest-scored branch accordingly. The security proof of our system is quite involving, and we consider it a significant technical contribution of this work. In a nutshell, security is demonstrated by analyzing honest users' local tree dynamics. We prove that, given any score distribution (due to the universal sampler of [5,16]), our protocol satisfies robust ledger properties with overwhelming probability.

Our contributions. They are threefold: (1) we abstract the process of generating transactions from bidding with a combinatorial optimization problem that extends the GMAP [21]; (2) we design a dedicated blockchain protocol for the P2PET system that can be further extended for general double-sided auction markets; (3) we prove that our protocol fulfills the ledger properties by modeling the computation in PoBA with the modified universal sampler [5,16].

1.3 Organization

The remainder is organized as follows. Section 2 defines the notation and the models of the protocol execution. The following three sections present our main contribution, *i.e.*, the dedicated blockchain protocol design for the P2PET system. Concretely, Section 3 introduces our tailored data structure and formally defines the BAP; Section 4 proposes the PoBA scheme based on the BAP and models the scheme with a universal sampler; and Section 5 describes the full protocol concerning blockchain selection and maintenance. Then, in Section 6, we prove the security of our protocol with respect to robust ledger properties, by analyzing users' local blockchain dynamics. Finally, Section 7 concludes this work and discusses potential extensions.

2 Preliminaries

In this paper, let λ be the security parameter. For a set X, $x \overset{\$}{\leftarrow} X$ denotes that x is uniformly and randomly sampled from 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 $[k] \overset{\Delta}{=} \{1, \ldots, k\}$, 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 SIG that satisfies correctness and existential unforgeability under adaptive chosen message attacks ($\mathsf{EUF\text{-}CMA}$) [14]. In general, the scheme consists of a tuple of algorithms $\mathsf{SIG} \overset{\Delta}{=} (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$.

- $\mathsf{KGen}(1^\lambda)$ takes as input the security parameter 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.

Additionally, we use $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^\lambda$ to denote a collision-free hash function. Next, we will describe the protocol execution model.

2.1 Protocol Execution Model

We model general protocol executions with the standard Interactive Turing Machines (ITM) Model approach [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. We first present our protocol settings.

Time and slots. Time is divided into discrete units called time slots, indexed by an integer $\ell \in \{1, 2, \dots\}$. We assume a globally synchronized clock $\mathcal{T}$ is equipped with a key pair $(\mathsf{sk}_\mathcal{T}, \mathsf{pk}_\mathcal{T})$ from the signature scheme SIG. Users can submit queries $(\sigma_\mathcal{T}, \ell) \leftarrow \mathcal{T}(m)$ such that $(\sigma_\mathcal{T}, \ell) \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_\mathcal{T}, m, \ell)$ where m is the query message and ℓ is the index of the current slot;

Synchrony. We adapt the δ-synchronous setting from [6] 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 received by all honest users in any slot $\ell \geq t + \delta$. Moreover, we assume the diffusion functionality from [12];

Rushing adversary. We consider a rushing network adversary who is able to: (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.

Permissionless setting with static corruption. We follow the constrained permissionless setting from [22]. In each slot, there are exactly $n \in \mathbb{N}$ users executing the protocol, and at least one is honest. Whenever an honest user joins, the protocol informs her with the parameters (n, δ). Moreover, we assume a static corruption model so that the adversary cannot corrupt honest users after they are spawned.

P2PET Rationale. Recall that in P2PET, users must periodically perform energy transmission and payment in real life. Hence, following the approach from [17], time is composed by time slots. Local clocks of P2PET users are implemented by certificated 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 be insignificant. Hence, it is convenient to assume the existence of the globally synchronized clock. Moreover, the certificated hardware naturally enables us to consider the permissioned setting, in which the $n \in \mathbb{N}$ users are initialized before the protocol execution and are informed with the identities of all honest ones. However, such a setting is not necessary in our security proof. Hence, we adopt the constrained permissionless setting as shown above.

3 Abstractions of Blockchain-Based P2PET

We present our redesigned blockchain protocol, data structure and considered problem, in this section. However, we start by providing an overview.

3.1 Intuition and Overview

The first step is to define the blockchain's data structure to support operations in the P2PET system. Recall that the system enables its users to buy or sell energy with energy storage and transmit equipment. Concretely, if a user produces more energy than she needs, the surplus can be stored. Then, the user can use the energy afterward or sell it through the system network. The buy/sell operation resembles a double-sided auction market with energy as trading resource. Hence, we borrow the term "bid", which is further categorized as "buy bid" (from buyers) or "sell bid" (sometimes called "ask", from sellers). Therefore, in order to support the bid operation, we add a bid layer to the conventional "transaction-block" structure. Moreover, as in conventional blockchain protocols, a transaction is regarded as the agreement of trading, which, in our case, is the energy transmission and payment agreement between a buyer and a seller. Thus, we define the transaction as a combination of a buy and a sell bid with some restrictions which are specified in later sections.

Next, we abstract the process of generating a set of transactions, from a given set of bids, as the early outlined BAP, in which we define a general scoring function to quantify the quality of the solutions, *i.e.*, each presented set of transactions. The BAP, with respect to the scoring function, is a combinatorial optimization problem, and can be regarded as a many-to-many assignment problem, a special case of the GMAP [21], as it was already described. Note that the BAP is the underlying problem of our "proof-of-X" for our blockchain protocol. We formally introduce the BAP and review the GMAP in the next sections.

3.2 Definitions for the Blockchain Data Structure

The redefined data structure definitions are bids, transactions, and, for completeness, block and chain. In the following, we denote each instance as bid, tx, bk, and chain, whereas we denote users as $\mathcal{U}$. For bid or block, when specifying issuer (or generator) $\mathcal{U}$ and time slot $\ell \geq 1$, we denote them explicitly as $\mathsf{bid}_{\mathcal{U}}^{\ell}$ or $\mathsf{bk}_{\mathcal{U}}^{\ell}$, respectively. Moreover, each instance is associated with an identifier, which is set to be the hash of the instance's contents, *e.g.*, $\mathsf{bidID} = \mathsf{Hash}(\mathsf{bid})$ is the identifier of a given bid bid. As we mentioned in the execution model, within the definitions, we assume that the global time server $\mathcal{T}$ is secured by the signature scheme SIG. The server $\mathcal{T}(m)$ returns queries for m, from users, with $(\sigma_{\mathcal{T}}, \ell) \leftarrow \mathcal{T}(m)$ such that $\sigma_{\mathcal{T}} = \mathsf{SIG.Sign}(\mathsf{sk}_{\mathcal{T}}, m, \ell)$ and time slot ℓ.

Starting with bids, users have 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 upper is acceptable). A bid should include its generation and expiration time slots, and must be signed by its issuer and the time server. The formal definition is as follows.

Definition 1 (Bid). *A bid* bid *issued by a user* $\mathcal{U}$ *who holds a key pair* $(\mathsf{sk}, \mathsf{pk})$ *from the signature scheme* SIG *is defined as* $\mathsf{bid} \stackrel{\Delta}{=} (\mathsf{bid}_{\mathsf{raw}}, \mathsf{aux}_{\mathcal{U}}, \mathsf{aux}_{\mathcal{T}})$ *and is associated with an identifier* $\mathsf{bidID} = \mathsf{Hash}(\mathsf{bid})$ *where:*

– *The bid's raw content,* $\mathsf{bid}_{\mathsf{raw}} \stackrel{\Delta}{=} (\mathsf{kind} \in \{\mathsf{buy}, \mathsf{sell}\}, q, p, t_{\mathsf{Gen}}, t_{\mathsf{Exp}})$*:*
  - $\mathsf{kind} \in \{\mathsf{buy}, \mathsf{sell}\}$ *indicates the bid's kind, i.e., a buy bid or a sell bid;*
  - $q, p > 0$ *denote the bid's quantity and unit price, respectively;*
  - $t_{\mathsf{Gen}}, t_{\mathsf{Exp}}$ *denote the bid's generation and expiration time slots, respectively.*

– *Information relates to the user's signature,* $\mathsf{aux}_{\mathcal{U}} \stackrel{\Delta}{=} ((\mathsf{pk}, \sigma), \mathsf{misc})$*:*
  - (pk, σ) *is the user's public key and signature, i.e.,* $\sigma = \mathsf{SIG.Sign}(\mathsf{sk}, \mathsf{bid}_{\mathsf{raw}})$*;*
  - misc *contains additional information from the user, e.g., a certificate by a trusted authority attesting the user's public key.*

– $\mathsf{aux}_{\mathcal{T}} \stackrel{\Delta}{=} (\mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}})$ *consists of the public key and signature from the time server* $\mathcal{T}$*, i.e.,* $\sigma_{\mathcal{T}} = \mathsf{SIG.Sign}(\mathsf{sk}_{\mathcal{T}}, (\mathsf{bid}_{\mathsf{raw}}, \mathsf{aux}_{\mathcal{U}}))$*.*

Moreover, we denote the bid space by $\mathbb{BID}$*.*

Next, a transaction combines a buy and a sell bid at selected quantity and price. Intuitively, the transaction's quantity should not exceed the original bids' quantity, and the price should be in the range of the original bids' prices. However, note that we enable our protocol to handle more complex bid assignments, the concrete restrictions will be presented in Section 4.

Definition 2 (Transaction). *A transaction* tx *is defined as* $\mathsf{tx} \stackrel{\Delta}{=} (\mathsf{bidID}_1, \mathsf{bidID}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}})$ *and is associated with an identifier* $\mathsf{txID} = \mathsf{Hash}(\mathsf{tx})$ *where:*

– $\mathsf{bidID}_1, \mathsf{bidID}_2$ *are the identifiers of two bids* $\mathsf{bid}_1, \mathsf{bid}_2$*. We may use* $\mathsf{bid}_1, \mathsf{bid}_2$ *directly for simplicity;*

– $q_{\mathsf{tx}} \geq 0, p_{\mathsf{tx}} > 0$ *denote the agreed quantity and unit price of the trade.*

Moreover, we denote the transaction space by $\mathbb{TX}$*.*

Here, for completeness, we also present the definition of a block. A block embeds both a bid set and a transaction set, in which the transaction set is derived from the bid set via the BAP. The block should also include a hash link pointing to its parent block. Similar to bids, the user who generates blocks need to secure the block with the signatures from herself and the time server.

Definition 3 (Block). *A block* bk *generated by a user* $\mathcal{U}$ *with* $(\mathsf{sk}, \mathsf{pk})$ *from* SIG *is defined as* $\mathsf{bk} \stackrel{\Delta}{=} (\mathsf{prevHash}, \mathsf{bk}_{\mathsf{raw}}, \mathsf{aux}_{\mathcal{U}}, \mathsf{aux}_{\mathcal{T}})$*. It is associated with an identifier* $\mathsf{bkID} = \mathsf{Hash}(\mathsf{bk})$ *and a score* S_{bk}*. In a block* bk*:*

– $\mathsf{prevHash}$ *denotes the identifier of the block's parent block, i.e., the hash of the block that* bk *intends to extend;*

– *The block's raw content,* $\mathsf{bk}_{\mathsf{raw}} \stackrel{\Delta}{=} (\mathsf{BIDs}, \mathsf{TXs}, t)$*:*

  - $\mathsf{BIDs} \neq \emptyset, \mathsf{TXs}$ *denote the set of bids and transactions embedded in the block;*
  - t *denotes the block's generation time slot.*
- *Information of signatures* $\mathsf{aux}_{\mathcal{U}} \overset{\Delta}{=} ((\mathsf{pk}, \sigma), \mathsf{misc})$ *and* $\mathsf{aux}_{\mathcal{T}} \overset{\Delta}{=} (\mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}})$ *are similar as in bids (Definition 1) where* $\sigma = \mathsf{SIG.Sign}(\mathsf{sk}, (\mathsf{prevHash}, \mathsf{bk}_{\mathsf{raw}}))$ *and* $\sigma_{\mathcal{T}} = \mathsf{SIG.Sign}(\mathsf{sk}_{\mathcal{T}}, (\mathsf{prevHash}, \mathsf{bk}_{\mathsf{raw}}, \mathsf{aux}_{\mathcal{U}}))$.

Moreover, we denote the block space by $\mathbb{BK}$.

Finally, as in conventional blockchain protocols, the chain is defined as an ordered linked list of blocks. The first block is called *genesis block* and denoted by bk^G, which contains public keys of users and is publicly known to all users. Then, all the users, when aware of new block candidates, will discard the ones which are signed with public keys that are not in the initial list.

Definition 4 (Chain). *Let* $||$ *denotes block concatenation, i.e., if* $\mathsf{bk}^t||\mathsf{bk}^{t+1}$, *then* bk^{t+1}*'s* $\mathsf{prevHash}$ *equals to* bk^t*'s identifier. Hence, a chain* chain *is an ordered linked list of blocks defined as* $\mathsf{chain} \overset{\Delta}{=} \mathsf{bk}^G||\mathsf{bk}^1||\mathsf{bk}^2||\cdots$ *where* bk^G *is the genesis block that contains the public keys of the users of the system.*

Note that we consider a tree structure in protocol description (Section 5) and security analysis (Section 6). Hence, we may use "branch", *i.e.*, branch, as an interchangeable term of "chain" (Definition 14).

3.3 The Bid Assignment Problem (BAP)

Based on the data structure above, we propose BAP that abstracts the process of finding an "optimal" set of combinations of bids, *i.e.*, a set of transactions. The optimality is defined based on a scoring function which should be sophisticatedly designed according to real-life situation, *e.g.*, market dynamics in the P2PET system. We emphasize again that this work focuses on the protocol level design instead of fully investigating the underlying energy trading market. Hence, we leave the scoring functions general to showcase the flexibility of our design.

Concretely, we proceed our definitions by first showing a generic framework of the BAP with an unspecified scoring function. Next, we consider the scoring function on the transaction level and then extend it to the blocks. These scoring functions are still general, but their domains are defined on the concrete data structure, *i.e.*, transaction sets and blocks. The scoring function for transaction sets is taken as an intermediate step, because a transaction set must be output before generating a block. Finally, with the scoring function for blocks, we define the BAP for block generation (bk-BAP). The purpose of bk-BAP is to argue our highest-score based block selection rules in the protocol design.

The Generic BAP. Our explanation starts from the generic version of the BAP, *i.e.*, we consider an unspecified scoring function $s : \mathbb{X} \to \mathbb{R}$ where $\mathbb{X}$ is an arbitrary space. Later, we instantiate s. The BAP takes as input a non-empty set of bids, denoted by BIDs. Here, for simplicity, we assume all bids in the set are "alive", *i.e.*, the bid was issued before and is not expired, and all bids' signatures are valid (the signature scheme's verification algorithm outputs 1, *i.e.*, $1 \leftarrow \mathsf{SIG.Verify}$). Hence, we can focus on the quantity and price constraints in the BAP. That is, we reframe the bids as $\mathsf{bid} = (\mathsf{kind}, q, p)$ in the following description. Next, according to each bid's kind, we can further divide the input set BIDs into a buy bid set and a sell bid set. Without loss generality, we write $\mathsf{BIDs} = B \cup S$ where $B \overset{\Delta}{=} \{\mathsf{bid}_i^B = (\mathsf{buy}, q_i^B, p_i^B)\}_{i \in [m]}$ is a buy bid set of size m, and $S \overset{\Delta}{=} \{\mathsf{bid}_j^S = (\mathsf{sell}, q_j^S, p_j^S)\}_{j \in [n]}$ is a sell bid set of size n such that $B \cap S = \emptyset$. Then, all the combinations of the bids can be given by the index set $I \overset{\Delta}{=} \{(i,j) : \mathsf{bid}_i^B \in B \wedge \mathsf{bid}_j^S \in S\} = \{(i,j)\}_{i \in [m], j \in [n]}$. An assignment between a buy bid $\mathsf{bid}_i^B \in B$ and a sell bid $\mathsf{bid}_j^S \in S$ is a transaction in the sense of Definition 2, and it can be denoted by a tuple with respect to indices: (i, j, q_{ij}, p_{ij}) where q_{ij} and p_{ij} are the assigned quantity and price, respectively.

Next, we consider the assignment constraints. Note that unlike matching problems, we enable many-to-many assignment, *i.e.*, a buy bid can be assigned to multiple sell bids, and multiple buy bids can be assigned to a sell bid. Hence, the total assigned quantity of a bid should not exceed the bid's original quantity, and the assigned price of a buy and a sell bid should fall in the range of their original prices. Then, for all $(i,j) \in I$, we have the following constraints.

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

We can now clarify the scoring function's domain by defining the assignment space $\mathbb{A}$. Concerning the input bid set $\mathsf{BIDs} \neq \emptyset$, denote an assignment set with $A \overset{\Delta}{=} \{(i, j, q_{ij}, p_{ij}) : \text{Given } \mathsf{BIDs}, \forall (i,j) \in I, \text{ Equation 1 holds}\}$. Then, the space of all assignment sets with respect to BIDs can be defined as $\mathbb{A}_{\mathsf{BIDs}}$. We define the space of all assignment sets as $\mathbb{A} \overset{\Delta}{=} \{A : A \in \mathbb{A}_{\mathsf{BIDs}}\}_{\mathsf{BIDs} \in \mathbb{BID}}$ and rewrite the scoring function as $s : \mathbb{A} \to \mathbb{R}$. Finally, the formal definition of our Generic BAP is given as follows.

Definition 5 (Generic BAP). *Let $\mathsf{BIDs} \subseteq \mathbb{BID}$ be a non-empty set of bids, which can be divided into $B = \{\mathsf{bid}_i^B = (\mathsf{buy}, q_i^B, p_i^B)\}_{i \in [m]}$ and $S = \{\mathsf{bid}_j^S = (\mathsf{sell}, q_j^S, p_j^S)\}_{j \in [n]}$, such that $\mathsf{BIDs} = B \cup S$ and $B \cap S = \emptyset$. Let $I = \{(i,j)\}_{i \in [m], j \in [n]}$ be the index set. Given the scoring function $s : \mathbb{A} \to \mathbb{R}$ where $\mathbb{A}$ denotes the assignment space, the Generic BAP is to find a set $X \overset{\Delta}{=} \{(i, j, q_{ij}, p_{ij}) : (i,j) \in I\}$ that maximizes $s(X)$ and satisfies the constraints given by Equation 1.*

We say a solution to the BAP is "valid" if the output set X is an assignment set, *i.e.*, $X = \{(i, j, q_{ij}, p_{ij}) : \text{Given } \mathsf{BIDs}, \forall (i,j) \in I, \text{ Equation 1 holds}\}$. Hence, the validity of BAP solutions does not require optimality, which is convenient to define the BAP-based "proof-of-X" scheme later in Section 4 (Definition 10).

BAP for Block Generation. The Generic BAP provides an intuition of the problem's input and output. However, in order to integrate the BAP into our

P2PET blockchain protocol, we need to adapt the problem so that it can output blocks, and the scoring function should be able to quantify the quality of blocks. As mentioned above, we first consider a general scoring function for transaction sets. Then, here, we extend it to the scoring function for blocks s_{bk}. The later will instantiate the scoring function in the Generic BAP (Definition 5) and complete our BAP for block generation (bk-BAP) Definition next.

We start from a function that evaluates one transaction $s_{\mathrm{tx}} : \mathbb{TX} \to \mathbb{R}$ and an aggregation function $\mathsf{Agg}_{\mathrm{txs}} : \mathbb{R}^* \to \mathbb{R}$. The aggregation function takes as input all evaluated values within the given transaction set and outputs the aggregated value as the score of the set. Hence, the scoring function for an arbitrary k-size transaction set $\mathsf{TXs} = \{\mathsf{tx}_i\}_{i\in[k]}$ can be written as $s_{\mathrm{txs}}(\mathsf{TXs}) \stackrel{\Delta}{=} \mathsf{Agg}_{\mathrm{txs}}(s_{\mathrm{tx}}(\mathsf{tx}_1), \ldots, s_{\mathrm{tx}}(\mathsf{tx}_k))$. Here, we show a toy example. Let $s_{\mathrm{tx}}(\mathsf{tx}) = q_{\mathsf{tx}} \cdot p_{\mathsf{tx}}$ for any transaction $\mathsf{tx} = (\mathsf{bid}_1, \mathsf{bid}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}}) \in \mathbb{TX}$, and let $\mathsf{Agg}_{\mathrm{txs}}$ be the sum function. Then, given any transaction set $\mathsf{TXs} \subseteq \mathbb{TX}$, $s_{\mathrm{txs}}(\mathsf{TXs}) = \sum_{\mathsf{tx}\in\mathsf{TXs}} q_{\mathsf{tx}} \cdot p_{\mathsf{tx}}$.

Next, recall the structure of blocks given in Definition 3. For simplicity, we reframe the blocks as $\mathsf{bk} = (\mathsf{TXs}, \mathsf{AUX})$ where $\mathsf{AUX} = (\mathsf{prevHash}, \mathsf{BIDs}, t, \mathsf{aux}_{\mathcal{U}}, \mathsf{aux}_{\mathcal{T}})$. Following the same process of defining s_{txs}, let s_{aux} be the scoring function for auxiliary information, and $\mathsf{Agg}_{\mathrm{bk}} : \mathbb{R}^2 \to \mathbb{R}$ be a general aggregation function. The function $\mathsf{Agg}_{\mathrm{bk}}$ aggregates the score of a transaction set and the score of auxiliary information within the given block. Hence, given any block $\mathsf{bk} = (\mathsf{TXs}, \mathsf{AUX}) \in \mathbb{BK}$, the scoring function for blocks is defined as $s_{\mathrm{bk}}(\mathsf{bk}) \stackrel{\Delta}{=} \mathsf{Agg}_{\mathrm{bk}}(s_{\mathrm{txs}}(\mathsf{TXs}), s_{\mathrm{aux}}(\mathsf{AUX}))$. We have the following definition.

Definition 6 (Scoring Function for Blocks). *Let $s_{tx} : \mathbb{TX} \to \mathbb{R}$ be a function that maps a transaction to a real value, and let $\mathsf{Agg}_{txs} : \mathbb{R}^* \to \mathbb{R}$ be a general aggregation function. The scoring function for transaction sets is $s_{txs} : \mathbb{TX}^* \to \mathbb{R}$ such that for any k-size transaction set $\mathsf{TXs} = \{\mathsf{tx}_i\}_{i\in[k]}$, then $s_{txs}(\mathsf{TXs}) \stackrel{\Delta}{=} \mathsf{Agg}_{txs}(s_{tx}(\mathsf{tx}_1), \ldots, s_{tx}(\mathsf{tx}_k))$. Let $s_{aux} : \{0,1\}^* \to \mathbb{R}$ be a function that maps auxiliary information of blocks to a real value. We use $\{0,1\}^*$ to denote the unspecified input domain. Next, let $\mathsf{Agg}_{bk} : \mathbb{R}^2 \to \mathbb{R}$ be another aggregation function that takes as input two real values. The scoring function for blocks is given by $s_{bk} : \mathbb{BK} \to \mathbb{R}$ such that for any block $\mathsf{bk} = (\mathsf{TXs}, \mathsf{AUX})$:*

$$s_{bk}(\mathsf{bk}) \stackrel{\Delta}{=} \mathsf{Agg}_{bk}(s_{txs}(\mathsf{TXs}), s_{aux}(\mathsf{AUX})).$$

Finally, we refine the Generic BAP with our newly defined scoring function for blocks s_{bk} to complete the bk-BAP.

Definition 7 (bk-BAP). *Let $\mathsf{BIDs} \subseteq \mathbb{BID}$ be a non-empty set of bids, which can be divided into $B = \{\mathsf{bid}_i^B = (\mathsf{buy}, q_i^B, p_i^B)\}_{i\in[m]}$ and $S = \{\mathsf{bid}_j^S = (\mathsf{sell}, q_j^S, p_j^S)\}_{j\in[n]}$, such that $\mathsf{BIDs} = B \cup S$ and $B \cap S = \emptyset$. Let $I = \{(i,j)\}_{i\in[m],j\in[n]}$ be the index set. Given the scoring function for blocks $s_{bk} : \mathbb{BK} \to \mathbb{R}$ as in Definition 6, the bk-BAP is to find a block $\mathsf{bk} = (\mathsf{TXs}, \mathsf{AUX})$ where $\mathsf{TXs} = \{(\mathsf{bid}_i^B, \mathsf{bid}_j^S, q_{ij}, p_{ij}) : (i,j) \in I\}$ and $\mathsf{BIDs} \in \mathsf{AUX}$. The block bk should maximize $s_{bk}(\cdot)$, and the transaction set TXs should satisfy constrains given by Equation 1.*

Remark 1 (Extensions). The data structure and BAPs can be extended in multiple ways depending on real-life requirements of the P2PET system. For example: When trading energy, users may have preference targets to sell to or buy from. Hence, we can embed a target list (potentially ordered according to priority) within the bid data structure, *i.e.*, $(\mathcal{U}, \mathsf{targets}) \in \mathsf{bid}$ where $\mathcal{U}$ is the bid's issuer, and $\mathsf{targets} \subseteq \{\mathcal{U}_i\}_{i\in[N]}$ is a subset of the all N users with whom the user willing to trade. Then, the BAPs should: (1) take into consideration the target constraints when assigning bids, *i.e.*, for any pair of buy and sell bids, $\mathsf{bid}_1 = (\mathsf{buy}, \mathcal{U}_1, \mathsf{targets}_1)$ and $\mathsf{bid}_2 = (\mathsf{sell}, \mathcal{U}_2, \mathsf{targets}_2)$, addition to constraints in Equation 1, assignment $(\mathsf{bid}_1, \mathsf{bid}_2)$ should also satisfy $\mathcal{U}_1 \in \mathsf{targets}_2 \wedge \mathcal{U}_2 \in \mathsf{targets}_1$; (2) adjust scoring function so that prioritized target grant higher scores. We hope this example can demonstrate our abstraction's capability of modeling the real-life system.

In the following two sections, we show our design of a blockchain protocol that takes the "proof-of-X" scheme based on the vanilla bk-BAP as its core. We argue that extensions like the example given above can be easily integrated into our design significant changes in the (yet to be presented) security analysis.

4 BAP-Based "Proof-of-X" Scheme

This section introduces a "proof-of-X" scheme based on the bk-BAP, namely, the Proof-of-Bid-Assignment (PoBA) scheme. Like conventional PoW, the PoBAP involves two types of participants: provers and verifiers. Note that both types are performed by users in our protocol. The separation here only aims to clarify the algorithms, *i.e.*, the prover runs a solving algorithm to solve the bk-BAP; Whereas the verifier evaluates the solution's validity and its score.

In order to present our scheme, we consider the setting where provers maintain a pool of bids alongside their views of the blockchain. For simplicity, we assume all bids are issued with valid signatures and correct identifiers, *i.e.*, computed honestly from the hash function. The "bidpool" is similar to the mempool in other blockchain protocols, with the difference that the mempool keeps transactions. A prover needs to update her bidpool according to the bidding history in the P2PET system and the transaction history recorded by the blockchain.

Our PoBA scheme starts with each prover holding a bidpool, and then the prover samples a bid set as the input for the bk-BAP. Hence, we first (1) show an algorithm for updating the bidpool, and then (2) present the formal syntax of the PoBA scheme. Finally, we model the scheme with a universal sampler [5,16], which has the interesting property of allowing random sampling from arbitrary distribution. In our case, we rely on this property to randomly sample PoBA solutions (blocks and corresponding scores). Moreover, we argue the reason and the limitation of this modeling.

We clarify that this section focuses on algorithms and the model of PoBA. PoBA-based block selection and blockchain maintenance are explained in Section 5, which utilizes our modeling, is presented in Section 6.

4.1 Bidpool Update

The purpose of the bidpool is to keep track of a continuous view of all available bids in each time slot, so that provers can sample their input bid sets for generating blocks by solving the bk-BAP. Each prover maintains her bidpool concerning two aspects: (1) The bidding and transaction history embedded in the prover's blockchain; (2) The newly issued bids in the previous slot. Therefore, this section starts with the definitions of the history. Then, by introducing the "residual" of bids, *i.e.*, the unassigned (quantity) part of bids in the history, we show the concrete approach and specify the algorithm for updating the bidpool (the yet to be introduced Algorithm 1) later.

Definition 8 (Bid History and Transaction History). *For any prover, let* $\mathsf{chain} = \mathsf{bk}^G || \mathsf{bk}^1 || \ldots || \mathsf{bk}^\ell$ *be the prover's blockchain, and for* $t \in [\ell]$*, let* $\mathsf{bk}^t_{\mathsf{raw}} = (\mathsf{BIDs}^t, \mathsf{tx}^t, t) \in \mathsf{bk}^t$*. The bid and transaction history with respect to* chain *is defined as* $\boldsymbol{H}^\ell_{bid} \triangleq \{\mathsf{bidID} : \mathsf{bid} \in \mathsf{BIDs}^1 \cup \cdots \cup \mathsf{BIDs}^\ell\}$ *and* $\boldsymbol{H}^\ell_{tx} \triangleq \{\mathsf{txID} : \mathsf{tx} \in \mathsf{TXs}^1 \cup \cdots \cup \mathsf{TXs}^\ell\}$ *where* bidID *and* txID *are the identifiers of* bid *and* tx*, respectively. For convenience, we may also use the corresponding bid or transaction for the given identifier.*

Recall that the transaction sets are generated following the constraints given in Equation 1, *i.e.*, the sum of assigned quantity in all transactions that involve a given bid should not surpass the bid's original quantity. Thus, in order to reduce the waste in energy trading, we enable provers to include bids with unassigned quantity into their bidpool. We name such bids as residual bids. A residual bid is a bid that exists on the blockchain, *i.e.*, in the bid history, that has unassigned quantity larger than 0.

Definition 9 (Residual Bid). *Let* $\mathsf{chain} = \mathsf{bk}^G || \mathsf{bk}^1 || \ldots || \mathsf{bk}^\ell$ *be a blockchain. Denote its bid and transaction history with* $\boldsymbol{H}^\ell_{bid}$ *and* $\boldsymbol{H}^\ell_{tx}$*, respectively. Given a bid* $\mathsf{bid} = (\mathsf{kind}, q, p, \mathsf{aux})$ *with identifier* $\mathsf{bidID} \in \boldsymbol{H}^\ell_{bid}$*, the residual bid of* bid *is defined as* $\mathsf{rbid} \triangleq (\mathsf{kind}, q_{\mathsf{rbid}}, p, \mathsf{aux})$ *if* $q_{\mathsf{rbid}} > 0$*, and* $\mathsf{rbid} \triangleq \perp$ *if* $q_{\mathsf{rbid}} \leq 0$*. Here:*

$$q_{\mathsf{rbid}} = q - \sum_{\{\mathsf{tx}:\mathsf{txID} \in \boldsymbol{H}^\ell_{tx} \wedge \mathsf{bidID} \in \mathsf{tx}\}} q_{\mathsf{tx}}. \tag{2}$$

If $\mathsf{rbid} \neq \perp$*, we set its identifier to* bidID*, i.e., the original bid's identifier. We denote the set of all residual bids with respect to a given chain as* R^{chain}*.*

Provers can assign a residual bid into a new transaction without exhausting the bid's q_{rbid}, thereby deriving a new residual bid from the original residual bid. That is, we enable provers to assign bids recursively as long as the bid is not expired. Moreover, because the residual bid's identifier is identical to its original (residual) bid, it is possible to maintain the history of each bid by tracking unique identifiers. This is also the reason we define the history using identifiers in Definition 8. Therefore, in the following, we denote the bidpool with Pool and use mappings to represent entries in the bidpool, *i.e.*, $(\mathsf{bidID}{:}\mathsf{bid}$ or $\mathsf{rbid}) \in \mathsf{Pool}$ where bidID is the identifier of a (residual) bid.

The algorithm for updating bidpools. Considering the process of updating the bidpool for any prover $\mathcal{P}$, at the beginning of time slot $\ell \geq 1$, let the prover hold a bidpool from the previous slot, denoted by $\mathsf{Pool}^{\ell-1}$, and a blockchain $\mathsf{chain}^{\ell-1} = \mathsf{bk}^{G}||\mathsf{bk}^{1}||\ldots||\mathsf{bk}^{\ell-1}$. We denote the set of bids in which all bids are issued in slot $\ell-1$ with $P^{\ell-1} \overset{\Delta}{=} \{\mathsf{bid} : t_{\mathsf{Gen}} = \ell-1\}$. Here, t_{Gen} is the bid's generation time slot. Then, the algorithm that outputs the updated bidpool for slot ℓ can be written as $\mathsf{Pool}^{\ell} \leftarrow \mathsf{UpdatePool}(\mathsf{Pool}^{\ell-1}, \mathsf{chain}^{\ell-1}, P^{\ell-1})$. For consistency of format, we may reframe the bid set P in the form of mapping, *i.e.*, $P^{\ell-1} = \{(\mathsf{bidID{:}bid}) : t_{\mathsf{Gen}} = \ell-1\}$. Note that $\ell = 1$ is slightly different because in the previous slot, $\mathsf{Pool}^{G} = \emptyset$ and $\mathsf{chain} = \mathsf{bk}^{G}$. That is, it contains no history, *i.e.*, $\boldsymbol{H}^{G}_{\text{bid}} = \emptyset, \boldsymbol{H}^{G}_{\text{tx}} = \emptyset$.

Next, we clarify that for any $t \in [\ell-1]$, the block bk^{t-1} is generated based on the bidpool in the same slot Pool^{t-1}. In other words, we need to remove the duplicated identifiers from the bidpool referring to the block's bid set. Then, by adding the set of freshly issued bids from the previous slot to the bidpool, we obtain a pool that contains all viable unique bid identifiers. Thus, we can refer to the transaction history on the blockchain to derive residual bids for these identifiers. The last step of the $\mathsf{UpdatePool}$ algorithm is to remove the outdated bids, *i.e.*, bids with expiration slots t_{Exp} earlier than the current slot. Finally, the algorithm outputs the new bidpool from the identifiers and their corresponding bids or residual bids. The procedure is formally specified in Algorithm 1.

4.2 Formal Syntax of PoBA

Given any time slot $\ell \geq 1$, the PoBA scheme consists of the tuple of algorithms $\mathsf{PoBA} \overset{\Delta}{=} (\mathsf{SampleBIDs}, \mathsf{Solve}, \mathsf{Eval})$. The $\mathsf{SampleBIDs}$ algorithm samples a bid set from the prover's updated bidpool as the input of the bk-BAP; Solve outputs the corresponding blockchain of a valid solution (block) to the bk-BAP. As mentioned before, by valid, we mean the solution satisfying the constraints and signatures being valid; Finally, the evaluation algorithm Eval verifies the validity of the whole blockchain and outputs the score of the latest block according to the public scoring function. We define the PoBA correctness (Definition 12) after presenting the formal syntax in the next definition.

Definition 10 (PoBA Scheme). *Let* $\mathsf{Hash} : \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda}$ *be a collision-free hash function, and let* $s_{bk} : \mathbb{BK} \rightarrow \mathbb{R}$ *be a publicly known scoring function for blocks as given in Definition 6. In time slot* $\ell \geq 1$*, for any prover* $\mathcal{P}$*, let* $\mathsf{Pool}^{\ell}_{\mathcal{P}}$ *be her updated bidpool from Algorithm 1, and let* $\mathsf{chain}^{\ell-1}_{\mathcal{P}} = \mathsf{bk}^{G}||\mathsf{bk}^{1}||\ldots||\mathsf{bk}^{\ell-1}$ *be her current blockchain. The prover performs* $(\mathsf{SampleBIDs}, \mathsf{Solve})$*, and any verifier performs* Eval*.*

– $\mathsf{SampleBIDs}(\mathsf{Pool}^{\ell}_{\mathcal{P}}, \mathsf{N}; r^{\ell}_{\mathcal{P}})$ *takes as input the prover's bidpool* $\mathsf{Pool}^{\ell}_{\mathcal{P}}$*, an upper bound* N *for the size of bid sets, and a random seed* $r^{\ell}_{\mathcal{P}}$*. The randomness can be omitted, we write it explicitly for later modeling the computation in PoBA.* $\mathsf{SampleBIDs}$ *outputs a set of bids* $\mathsf{BIDs}^{\ell}_{\mathcal{P}} \subseteq \mathsf{Pool}^{\ell}_{\mathcal{P}}$ *such that* $|\mathsf{BIDs}^{\ell}_{\mathcal{P}}| \leq \mathsf{N}$*;*

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

```
1  function UpdatePool($\mathsf{Pool}^{\ell-1}, \mathsf{chain}^{\ell-1}, P^{\ell-1}$);
2  Let $\mathsf{Pool}^{\ell} = \emptyset$;
3  if $\ell = 1$ then
4      Parse $\mathsf{Pool}^{G} = \emptyset$, $\mathsf{chain} = \mathsf{bk}^{G}$, and $P^{G} = \{(\mathsf{bidID{:}bid}) : t_{\mathsf{Gen}} = G\}$;
5      Return $\mathsf{Pool}^{1} = P^{G}$
6  else
7      Parse $\mathsf{chain}^{\ell-1} = \mathsf{bk}^{G}||\mathsf{bk}^{1}||\ldots||\mathsf{bk}^{\ell-1}$ and
        $\mathsf{bk}_{\mathsf{raw}}^{\ell-1} = (\mathsf{BIDs}^{\ell-1}, \mathsf{tx}^{\ell-1}, \ell{-}1) \in \mathsf{bk}^{\ell-1}$;
8      Parse $P^{\ell-1} = \{(\mathsf{bidID{:}bid}) : t_{\mathsf{Gen}} = \ell{-}1\}$;
       // Remove duplicated identifiers from $\mathsf{Pool}^{\ell-1}$.
9      for $\mathsf{bid} \in \mathsf{BIDs}^{\ell-1}$ do
10         if $\mathsf{bidID} \in \mathsf{Pool}^{\ell-1}$ then
11             Delete the entry from $\mathsf{Pool}^{\ell-1}$
12         end
13     end
       // Collect all unique bid identifiers.
14     Set $\mathsf{Pool}^{\ell} = \mathsf{Pool}^{\ell-1} \cup P^{\ell-1}$;
       // Derive (residual) bids for each identifier with Equation 2.
15     Parse the transaction history of $\mathsf{chain}$ as $\boldsymbol{H}_{\mathrm{tx}}^{\ell-1}$;
16     for $\mathsf{bidID} \in \mathsf{Pool}^{\ell}$ do
           // Let $q$ be the original bid's quantity with $\mathsf{bidID}$.
17         Compute $q_{\mathsf{rbid}} = q - \sum_{\{\mathsf{tx}:\mathsf{txID} \in \boldsymbol{H}_{\mathrm{tx}}^{\ell-1} \wedge \mathsf{bidID} \in \mathsf{tx}\}} q_{\mathsf{tx}}$;
18         if $q_{\mathsf{rbid}} > 0$ then
19             Replace the entry in $\mathsf{Pool}^{\ell}$ with $(\mathsf{bidID{:}rbid})$ such that $q_{\mathsf{rbid}} \in \mathsf{rbid}$;
20         end
21     end
       // Remove outdated bids from $\mathsf{Pool}^{\ell}$.
22     for $\mathsf{bidID} \in \mathsf{Pool}^{\ell}$ do
           // Let $t_{\mathsf{Exp}}$ be the bid's expiration slot with $\mathsf{bidID}$.
23         if $t_{\mathsf{Exp}} < \ell$ then
24             Delete the entry from $\mathsf{Pool}^{\ell}$
25         end
26     end
27     Return $\mathsf{Pool}^{\ell}$
28 end
```

– $\mathsf{Solve}(\mathsf{chain}_{\mathcal{P}}^{\ell-1}, \mathsf{BIDs}_{\mathcal{P}}^{\ell})$ *takes as input a bid set* $\mathsf{BIDs}_{\mathcal{P}}^{\ell}$ *that satisfies* $|\mathsf{BIDs}_{\mathcal{P}}^{\ell}| \leq \mathsf{N}$. *The algorithm outputs the prover's solution to the bk-BAP, i.e., a block candidate* $\mathsf{bk}_{\mathcal{P}}^{\ell} = (\mathsf{TXs}, \mathsf{AUX})$, *with the corresponding new blockchain* $\mathsf{chain}_{\mathcal{P}}^{\ell} \triangleq \mathsf{chain}_{\mathcal{P}}^{\ell-1}||\mathsf{bk}_{\mathcal{P}}^{\ell}$. *Here,* $\mathsf{AUX} = (\mathsf{prevHash}, \mathsf{bid}_{\mathcal{P}}^{\ell}, \ell, \mathsf{aux}_{\mathcal{P}}, \mathsf{aux}_{\mathcal{T}})$, $\mathsf{prevHash} = \mathsf{Hash}(\mathsf{bk}^{\ell-1})$, *and all signatures in* $(\mathsf{aux}_{\mathcal{P}}, \mathsf{aux}_{\mathcal{T}})$ *are valid;*

– $\mathsf{Eval}(\mathsf{chain}^t_{\mathcal{P}^*}, \mathsf{N})$ *takes as input a blockchain* $\mathsf{chain}^t_{\mathcal{P}^*}$ *from prover* $\mathcal{P}^*$ *and the size bound* N *for bid sets. If* $t \neq \ell$, Eval *outputs* $(0, \perp)$. *Otherwise, parse* $\mathsf{chain}^\ell_{\mathcal{P}^*} = \mathsf{chain}^{\ell-1} || \mathsf{bk}^\ell_{\mathcal{P}^*}$ *where* $\mathsf{bk}^\ell_{\mathcal{P}^*}$ *is generated by* $\mathcal{P}^*$, *and assume* $(1, \cdot) \leftarrow \mathsf{Eval}(\mathsf{chain}^{\ell-1}, \mathsf{N})$. *Let* $\boldsymbol{H}^{\ell-1}_{bid}$ *and* $\boldsymbol{H}^{\ell-1}_{tx}$ *denote the bid and transaction history of* $\mathsf{chain}^{\ell-1}$, *respectively. Parse* $\mathsf{bk}^\ell_{\mathcal{P}^*} = (\mathsf{prevHash}, (\mathsf{BIDs}^*, \mathsf{TXs}^*, t^*), \mathsf{aux}_{\mathcal{P}^*}, \mathsf{aux}_{\mathcal{T}})$, *if all the following conditions hold, the algorithm outputs* $(1, s_{bk}(\mathsf{bk}^\ell_{\mathcal{P}^*}))$, *and we say the blockchain and the new block are valid:*
  1. *For previous hash* $\mathsf{prevHash}$*: Let* $\mathsf{bk}^{\ell-1}$ *be the latest block on the verified blockchain* $\mathsf{chain}^{\ell-1}$*:* $\mathsf{prevHash} = \mathsf{Hash}(\mathsf{bk}^{\ell-1})$*;*
  2. *For bid set* bid^**: (1)* $|\mathsf{BIDs}^*| \leq \mathsf{N}$*; (2) For each* $\mathsf{bid} \in \mathsf{BIDs}^*$ *with generation and expiration time slots* $(t_{\mathsf{Gen}}, t_{\mathsf{Exp}})$, *the current slot* $\ell \in [t_{\mathsf{Gen}}, t_{\mathsf{Exp}}]$*; (3) Let* bidID *be* $\mathsf{bid} \in \mathsf{BIDs}^*$*'s identifier, if* $\mathsf{bidID} \notin \boldsymbol{H}^{\ell-1}_{bid}$, *and the signatures in* bid *are valid;*
  3. *For transaction set* TXs^**: Let* $BI = \{\mathsf{bidID} : \mathsf{bid} \in \mathsf{BIDs}^*\}$ *and* $TI = \{\mathsf{txID} : \mathsf{tx} \in \mathsf{TXs}^*\}$ *be the identifier sets given by* BIDs^* *and* TXs^*, *respectively. Then: (1) For each* $\mathsf{tx} = (\mathsf{bidID}_1, \mathsf{bidID}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}}) \in \mathsf{TXs}^*$, $\mathsf{bidID}_1, \mathsf{bidID}_2 \in \boldsymbol{H}^{\ell-1}_{bid} \cup BI$*; (2) Let* $\boldsymbol{H}^\ell_{tx,\mathcal{P}^*} = \boldsymbol{H}^{\ell-1}_{tx} \cup TI$. *For each* $\mathsf{tx} = (\mathsf{bidID}_1, \mathsf{bidID}_2, q_{\mathsf{tx}}, p_{\mathsf{tx}}) \in \boldsymbol{H}^\ell_{tx,\mathcal{P}^*}$, *without loss of generality, let* $\mathsf{bid}_{\mathsf{1raw}} = (\mathsf{buy}, q_1, p_1)$ *has identifier* bidID_1, *and* $\mathsf{bid}_{\mathsf{2raw}} = (\mathsf{sell}, q_2, p_2)$ *has identifier* bidID_2, *and the assigned quantity and price of* tx *satisfies:*

$$\sum_{\mathsf{txID} \in \boldsymbol{H}^\ell_{tx,\mathcal{P}^*} \wedge \mathsf{bidID}_1 \in \mathsf{tx}} q_{\mathsf{tx}} \leq q_1 \bigwedge \sum_{\mathsf{txID} \in \boldsymbol{H}^\ell_{tx,\mathcal{P}^*} \wedge \mathsf{bidID}_2 \in \mathsf{tx}} q_{\mathsf{tx}} \leq q_2 \bigwedge p_2 \leq p_{\mathsf{tx}} \leq p_1.$$

  4. *For block generation slot* t^**:* $t^* = \ell$*;*
  5. *For auxiliary information* $(\mathsf{aux}_{\mathcal{P}^*}, \mathsf{aux}_{\mathcal{T}})$*: The signatures are valid.*

  Otherwise, the algorithm outputs $(0, \perp)$.

Correctness of the PoBA scheme requires that the Eval algorithm accepts any blockchain from honestly executed $\mathsf{SampleBIDs}$ and Solve algorithms. Because the $\mathsf{SampleBIDs}$ algorithm starts with a bidpool, we need to first define validity for bidpools. Here, we consider a $\mathsf{VerifyPool}$ algorithm to check "conflicts" between an updated bidpool and the blockchain.

Definition 11 (Validity of Bidpool). *For any prover in time slot* $\ell \geq 1$, *let* Pool^ℓ *be her bidpool, and let* $\mathsf{chain}^{\ell-1} = \mathsf{bk}^G || \mathsf{bk}^1 || \ldots || \mathsf{bk}^{\ell-1}$ *be her blockchain. The prover performs* $\mathsf{VerifyPool}$.

– $\mathsf{VerifyPool}(\mathsf{Pool}^\ell, \mathsf{chain}^{\ell-1})$ *takes as input the bidpool* Pool^ℓ *and the blockchain* $\mathsf{chain}^{\ell-1}$. *Let* $\boldsymbol{H}^{\ell-1}_{bid}$ *and* $\boldsymbol{H}^{\ell-1}_{tx}$ *denote the history of* $\mathsf{chain}^{\ell-1}$. *If the following conditions hold,* $\mathsf{VerifyPool}$ *outputs* 1, *and we say the bidpool* Pool *is valid regarding the blockchain* chain.
  1. *The blockchain is valid, i.e.,* $(1, \cdot) \leftarrow \mathsf{PoBA.Eval}(\mathsf{chain}^{\ell-1}, \mathsf{N})$ *where* N *is the size bound of the bid set embedded in each block on the blockchain;*
  2. *For each* $\mathsf{bidID} \in \mathsf{Pool}^\ell$, *let its corresponding (residual) bid have quantity* q_{rbid}, *and have generation and expiration time slots* $(t_{\mathsf{Gen}}, t_{\mathsf{Exp}})$*: (1) The current time slot* $\ell \in [t_{\mathsf{Gen}}, t_{\mathsf{Exp}}]$*; (2) If* $\mathsf{bidID} \in \boldsymbol{H}^{\ell-1}_{bid}$, q_{rbid} *computed from*

Equation 2 is larger than 0*; (3) If* $\mathsf{bidID} \notin \boldsymbol{H}_{bid}^{\ell-1}$*, and the signatures in the corresponding bid are valid.*

Otherwise, the algorithm outputs 0*.*

The reason why $\mathsf{VerifyPool}$ considers only an updated bidpool regarding the blockchain is that provers would not keep tracking old bidpools and bid sets after updating them to the new bidpool. Hence, the proposed algorithm is to verify the validity of bidpools, instead of deciding the *correctness* of the $\mathsf{UpdatePool}$ algorithm. Next, we define the correctness of the PoBA scheme.

Definition 12 (Correctness of the PoBA Scheme). *Given any prover* $\mathcal{P}$ *in time slot* $\ell \geq 1$*, let* $(\mathsf{chain}^{\ell-1}, \mathsf{Pool}^{\ell}, \mathsf{N})$ *be the prover's input tuple such that* $(1, \cdot) \leftarrow \mathsf{Eval}(\mathsf{chain}^{\ell-1}, \mathsf{N})$ *and* $1 \leftarrow \mathsf{VerifyPool}(\mathsf{Pool}^{\ell}, \mathsf{chain}^{\ell-1})$*. The PoBA scheme is correct, if* $\mathsf{BIDs} \leftarrow \mathsf{SampleBIDs}(\mathsf{Pool}^{\ell}, \mathsf{N})$ *and* $\mathsf{chain}_{\mathcal{P}}^{\ell-1}||\mathsf{bk}_{\mathcal{P}}^{\ell} \leftarrow \mathsf{Solve}(\mathsf{chain}_{\mathcal{P}}^{\ell-1}, \mathsf{BIDs})$ *are honestly executed, then:*

$$\Pr\left[(1, s_{\mathsf{bk}}(\mathsf{bk}_{\mathcal{P}}^{\ell})) \leftarrow \mathsf{Eval}(\mathsf{chain}_{\mathcal{P}}^{\ell-1}||\mathsf{bk}_{\mathcal{P}}^{\ell}, \mathsf{N})\right] = 1.$$

Usually, in "proof-of-X" schemes that involve computational tasks, *e.g.*, PoW [7] and proof-of-useful-work [4, 9], the difficulty is another crucial property. Intuitively, it requires provers to contribute enough computing power to generate *valid* blocks. Otherwise, adversarial provers can generate massive blocks in a short period of time, and the network cannot be finalized on a chain of blocks but with many "forks".

However, the validity of blocks in our PoBA scheme does not require optimality. Hence, it is easy to generate valid blocks for any prover. Instead, our "difficulty" lies in the competition, *i.e.*, honest provers only select the *highest-scored* blocks (or precisely, blockchains as it will be introduced in Section 5.2). Relying on block scores in PoBA is meaningful because the score relates to the underlying P2PET system. Then, a higher-scored block is preferable to the system regardless of who (honest or not) generates the block. Hence, despite that we follow conventional modeling approaches [12] that differentiate the computing power of the honest provers in contrast with adversarial provers, we show in Section 6, that this differentiation does not change the security of our protocol (a higher score block, even if adversarial, is still useful for the overall system).

4.3 Modeling the Scheme

Recall that in PoW-based blockchain protocols, the hash computation is modeled by the number of queries to a random oracle. Our modeling follows a similar approach for PoBA, with respect to the queries to a modified universal sampler [5], which can be regarded as an advanced random oracle that samples from an arbitrary distribution. In our case, the distribution is determined by the general scoring function. The rationale behind it is to abstract away from concrete implementations of the general scoring function. We enable provers to obtain solutions of PoBA, *i.e.*, blocks and corresponding scores, by querying the universal sampler instead of running concrete algorithms. The modeling is appropriate based on the following evidence: (1) Given a well-chosen scoring function, the BAPs (generic and bk-BAP) can be reduced to the generalized multiple-assignment problem, which has been proven to be NP-complete [21]; (2) Stochasticity arises in assignment problems due to the uncertainty in problem inputs [8], *i.e.*, in our case, PoBA requires provers to sample input bid sets from bidpools for the bk-BAP, hence, some bids may not need to be assigned.

Next, we first show the definition of the modified universal sampler [5]. Then, we detail the interaction between PoBA provers and the universal sampler.

Definition 13 (Universal Sampler [5]). *A universal sampler scheme consists of algorithms* $\mathsf{US} \overset{\Delta}{=} (\mathsf{Setup}, \mathsf{Sample})$ *that are performed as follows.*

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

We instantiate the definition above by specifying the program description with $\mathsf{d} \overset{\Delta}{=} (\text{bk-BAP}, s_{\text{bk}})$ where bk-BAP is the underlying problem of the PoBA scheme and s_{bk} is the general scoring function associated with the bk-BAP. The universal sampler is accessible by any prover performing the PoBA scheme via queries in which the prover sends its own bidpool and a random seed. That is, for any prover $\mathcal{P}$, her query to the universal sampler is $\beta \overset{\Delta}{=} (\mathsf{Pool}_{\mathcal{P}}, r_{\mathcal{P}})$. Here, we follow the conventional model approach that in each time slot, each honest prover can make at most $q > 0$ queries, whereas, the adversarial prover can make at most $q_{\mathcal{A}} > q$ queries. We denote the upper bound of total query number in each slot by $Q \in \mathbb{N}$. The difference in query capabilities indicates the difference in computing power between honest and adversarial provers. Moreover, we clarify that the communication between provers and the universal sampler cannot be delayed by the *network* adversary, given it is *oracle access*. This is a natural setting since the universal sampler captures the capability of provers to internally and locally select bids and generate blocks relying only on her randomness and the solving algorithm.

Then, the Sample algorithm has the single property of randomly sampling a block $\mathsf{bk} \in \mathbb{BK}$ such that $s_{\text{bk}}(\mathsf{bk}) \overset{\mathcal{D}}{\leftarrow} \mathcal{S}$ where $\mathcal{D}$ and $\mathcal{S}$ denotes the score distribution and score space determined by the scoring function. With a well-chosen scoring function, blocks can be strictly ordered by the score with high probability. Concretely, the universal sampler works as follows.

Universal Sampler

In any time slot $\ell \geq 1$, setup up the universal sampler with $U \leftarrow \mathsf{US}.\mathsf{Setup}(1^\lambda)$. Let $\mathbb{L}^{\ell}_{\mathcal{P}} = \{(\cdot,\cdot,\cdot,\cdot)\} \in U$ be the list kept by the universal sampler for any prover $\mathcal{P}$. The total size of lists is upper bounded by $Q \in \mathbb{N}$, *i.e.*, let $\mathbf{P}^{\ell}$ denote the set of all provers in slot ℓ, $|\bigcup_{\mathcal{P} \in \mathbf{P}^{\ell}} \mathbb{L}^{\ell}_{\mathcal{P}}| \leq Q$.

On a query $(\mathsf{Pool}^{\ell}_{\mathcal{P}}, r^{\ell}_{\mathcal{P}})$ **from** $\mathcal{P}$**:**

- If: there exists a tuple $(\mathsf{Pool}^{\ell}_{\mathcal{P}}, r^{\ell}_{\mathcal{P}}, \mathsf{bk}^{\ell}_{\mathcal{P}}, s_{\mathrm{bk}}(\mathsf{bk}^{\ell}_{\mathcal{P}})) \in \mathbb{L}^{\ell}_{\mathcal{P}}$, then, return $(\mathsf{bk}^{\ell}_{\mathcal{P}}, s_{\mathrm{bk}}(\mathsf{bk}^{\ell}_{\mathcal{P}}))$;
- Else if: $|\mathbb{L}^{\ell}_{\mathcal{P}}| > q$ when $\mathcal{P}$ is honest or $|\mathbb{L}^{\ell}_{\mathcal{P}}| > q_{\mathcal{A}}$ when $\mathcal{P}$ is adversarial, then, return $\perp$;
- Else: run and return $(\mathsf{bk}^{\ell}_{\mathcal{P}}, s_{\mathrm{bk}}(\mathsf{bk}^{\ell}_{\mathcal{P}})) \leftarrow \mathsf{US.Sample}(U, (\text{bk-BAP}, s_{\mathrm{bk}}), (\mathsf{Pool}^{\ell}_{\mathcal{P}}, r^{\ell}_{\mathcal{P}}))$, and add $(\mathsf{Pool}^{\ell}_{\mathcal{P}}, r^{\ell}_{\mathcal{P}}, \mathsf{bk}^{\ell}_{\mathcal{P}}, s_{\mathrm{bk}}(\mathsf{bk}^{\ell}_{\mathcal{P}}))$ to $\mathbb{L}^{\ell}_{\mathcal{P}}$.

The output distribution. The output from the universal sampler can be modeled as a continuous random variable X following a score distribution $\mathcal{D}$ on a score space $\mathcal{S} \overset{\Delta}{=} [\mathsf{smin}, \mathsf{smax}]$. Here, $\mathcal{D}, \mathsf{smin}, \mathsf{smax}$ are determined by the general scoring function $s_{\mathrm{bk}}(\cdot)$. We denote the probability density function and the distribution function of $\mathcal{D}$ with $f(\cdot)$ and $F(\cdot)$ such that $F(x) = \Pr[X \leq x] = \int_{\mathsf{smin}}^{x} f(t)dt$.

Discussion: Block re-using attack. An issue with the PoBA scheme definition and our modeling is that it cannot prevent adversaries from re-using other users' valid block candidates, *e.g.*, the adversaries can claim others' blocks as theirs or modify the block slightly to achieve higher scores without performing enough computation. Hence, in order to tackle the block re-using attack, we consider an *exact* time barrier for diffusing block candidates in each time slot so that no honest user will diffuse its block before this time barrier. This is achievable given the globally synchronized clock $\mathcal{T}$. We clarify that this is the only place in this paper where we use the strong synchronicity of the global clock, and it is a natural setting for P2PET.

5 PoBA-Based Blockchain Protocol

As mentioned before, each user in our blockchain protocol maintains a bidpool alongside a blockchain. In the previous section, we show the process of the user maintaining her bidpool with Algorithm 1. We then introduce the PoBA scheme in which users can generate blocks by solving the bk-BAP (Definition 7 that takes as input a bid set sampled from the bidpool. This section ends the description of our protocol by further showing the process of users selecting blocks and maintaining their blockchains. Finally, we will briefly discuss the incentive model.

Concretely, in any given time slot, we continue our explanation from where each user obtains a list of block candidates from the PoBA scheme, which is modeled by our bk-universal sampler. Each honest user diffuses her highest-scored candidates through the network; Whereas, a rushing adversary, as mentioned in Section 2, receives all honest blocks, manages the order, and diffuses its (the adversary's) candidates accordingly. Since there is more than one block being delivered to users, and each block can only extend one blockchain, we consider a directed tree (or precisely, a directed forest due to missing blocks) structure that stores blocks locally for each user. The root of the "block-tree" is the genesis block as given in Definition 4. Users extend their block-tree in each time slot with newly received blocks and select the "best" branch on the block-tree as their blockchain. Hence, we extend the notion score for branches to support this selecting operation. Users will output the confirmed part of their blockchain when asked to report the ledger.

5.1 Block-Tree and Score of Branches

Now, we start with the necessary definitions for the block-tree structure and the scoring function for branches. Given a time slot $\ell \geq 1$, we first consider a master-tree mtree^ℓ that contains all *valid* blocks (block candidates) generated (*NOT* diffused) by users (honest or not) from the genesis slot to slot ℓ. Then, we define the master-tree $\mathsf{mtree}^\ell = (V, E)$ as a directed tree such that its vertex set corresponds to blocks and the edge set corresponds to the hash link between blocks. Hence, the genesis block bk^G is the root of mtree^ℓ. Recall the height definition from graph theory: (1) The vertex height in a directed tree is defined as the number of edges between the vertex and the root; (2) The tree height is defined as the number of edges in the longest path between a leaf vertex and the root. Hence, mtree^ℓ is of height ℓ. Furthermore, for vertices in mtree^ℓ of the same height, the corresponding blocks are generated in the same time slot. A user may only see a part of the master-tree because we assume the rushing adversary controls block diffusion. Hence, denote the block-tree of a user $\mathcal{U}$ as $\mathsf{tree}^\ell_\mathcal{U} = (V_\mathcal{U}, E_\mathcal{U})$, we have $\mathsf{tree}^\ell_\mathcal{U} \subseteq \mathsf{mtree}^\ell$, *i.e.*, $V_\mathcal{U} \subseteq V$ and $E_\mathcal{U} \subseteq E$. The formal definition is as follows.

Definition 14 (Master-Tree, User's Block-Tree, Branch). *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, let $BK_i^t \overset{\Delta}{=} \{\mathsf{bk}_i^t\} \neq \emptyset$ denote the set of valid blocks generated by user $\mathcal{U}_i$ in slot $t \in [\ell]$. Then, $BK^t \overset{\Delta}{=} \bigcup_{i\in[n]} BK_i^t$ denotes the set of valid blocks generated in slot t. The master-tree of slot ℓ is defined as $\mathsf{mtree}^\ell = (V, E)$ such that $V = \{\mathsf{bk}^G\} \cup \bigcup_{t\in[\ell]} BK^t$, and $E = \{(\mathsf{bk}^G, \mathsf{bk}^1) : \forall \mathsf{bk}^1 \in BK^1\} \cup \bigcup_{t\in[\ell-1]} \{(\mathsf{bk}^t, \mathsf{bk}^{t+1}) : \forall \mathsf{bk}^t \in BK^t, \mathsf{bk}^{t+1} \in BK^{t+1}, \mathsf{prevHash} = \mathsf{Hash}(\mathsf{bk}^t)\}$ where $\mathsf{prevHash}$ is the previous hash value entry in block bk^{t+1}, i.e., blocks $(\mathsf{bk}^t, \mathsf{bk}^{t+1})$ are linked by the hash function Hash as in the PoBA scheme (Definition 10). A block-tree of user $\mathcal{U}$ is denoted by $\mathsf{tree}^\ell_\mathcal{U} = (V_\mathcal{U}, E_\mathcal{U})$, and satisfies $\mathsf{tree}^\ell_\mathcal{U} \subseteq \mathsf{mtree}^\ell$. Moreover, given a block-tree (master or user's) of slot ℓ as $G^\ell = (V^\ell, E^\ell)$, a branch is defined $\mathsf{branch}^\ell \overset{\Delta}{=} \mathsf{bk}^G || \mathsf{bk}^1_{i_1} || \ldots || \mathsf{bk}^\ell_{i_\ell}$ where $I \overset{\Delta}{=} \{i_1, \ldots, i_\ell\}$ is the index set of block generators such that $\mathsf{branch}^\ell \subseteq G^\ell$, i.e., for any $t \in [\ell]$, $\mathsf{bk}^t_{i_t} \in V$, and for any $t \in [\ell-1]$ and $i_t, i_{t+1} \in I$, $(\mathsf{bk}^t_{i_t}, \mathsf{bk}^{t+1}_{i_{t+1}}) \in E$.*

Recall that the branch definition resembles the definition of blockchain as mentioned after Definition 4. We may also distinguish them by using branch for arbitrary branches on a given block-tree, and chain for the highest-scored branch.

Moreover, we use $\mathsf{branch}^{\ell\lceil k}$ for $k \in \mathbb{N}$ to denote the chain of blocks resulting from the removal of the k rightmost blocks of the branch branch^ℓ. If $k \geq \ell$, we define $\mathsf{branch}^{\ell\lceil k} = \varepsilon$, *i.e.*, the empty chain. Then, by $\mathsf{tree}^{\ell\lceil k}$, we denote the sub-tree constructing from $\mathsf{branch}^{\ell\lceil k}$ for all $\mathsf{branch} \subseteq \mathsf{tree}^{\ell}$. Note that in the case of master-tree mtree^{ℓ}, since it contains all generated blocks in the protocol, we have for any $t \in [\ell]$ and $k = \ell - t$: $\mathsf{mtree}^{t} = \mathsf{mtree}^{\ell\lceil k}$.

Next, in order to define the best branch with respect to scores, we extend the scoring function by taking blocks' generation slots into consideration. That is, given a branch $\mathsf{branch}^{\ell} = \mathsf{bk}^{G}||\mathsf{bk}^{1}||\ldots||\mathsf{bk}^{\ell}$ in time slot $\ell \geq 1$, we introduce an accumulating function $\mathsf{acc}(t) \in \mathbb{R}$ for all $t \in [\ell]$. Then, the score of each slot t is defined as $\mathsf{acc}(t) \cdot s_{\mathrm{bk}}(\mathsf{bk}^{t})$. Finally, the score of the branch is defined as the sum of all slot scores. Formally, we write the score of the branch $\mathsf{branch}^{\ell} = \mathsf{bk}^{G}||\mathsf{bk}^{1}||\ldots||\mathsf{bk}^{\ell}$ as follows.

$$S_{\mathsf{branch}^{\ell}} \triangleq \sum_{t=1}^{\ell} \mathsf{acc}(t) \cdot s_{\mathrm{bk}}(\mathsf{bk}^{t}). \tag{3}$$

Note that in our protocol, blocks generated in the same time slot share the same height in the block-tree. Then, given an arbitrary block-tree, users can compute the score of all branches on the tree with Equation 3. Hence, for a user $\mathcal{U}$ holding a block-tree $\mathsf{tree}^{\ell}_{\mathcal{U}}$ in time slot $\ell \geq 1$, the user selects her branch, denoted by $\mathsf{chain}^{\ell}_{\mathcal{U}} \subseteq \mathsf{tree}^{\ell}_{\mathcal{U}}$, such that:

$$S_{\mathsf{chain}^{\ell}_{\mathcal{U}}} = \max_{\mathsf{branch} \subseteq \mathsf{tree}^{\ell}_{\mathcal{U}}} S_{\mathsf{branch}}. \tag{4}$$

As we will show in Section 6, the accumulating function parameterizes the possibility of our protocol achieving consensus.

5.2 Protocol Description

Finally, we can present the full description of our protocol. In the following, we show the workflow of an honest user in an arbitrary time slot where she maintains her bidpool, generates a block, extends her block-tree, and reports her confirmed blockchain to the system.

Let $\mathcal{U}$ be an honest user among the $n \in \mathbb{N}$ users performing the protocol in time slot $\ell \geq 1$. Denote the user's view of bidpool and block-tree at the end of slot $\ell{-}1$ with $\mathsf{Pool}^{\ell-1}$ and $\mathsf{tree}^{\ell-1}$, respectively. Here, we do not specify them to $\mathcal{U}$ because our permissionless setting cannot guarantee the user to participate in slot $\ell{-}1$. However, we require that for any branch $\mathsf{branch} \subseteq \mathsf{tree}^{\ell-1}$, $(1, \cdot) \leftarrow \mathsf{PoBA.Eval}(\mathsf{branch}, \mathsf{N})$ as given in Definition 10. Then, following Equation 4, the user selects her branch, which is denoted by $\mathsf{chain}^{\ell-1}_{\mathcal{U}}$. Let $P^{\ell-1} \triangleq \{\mathsf{bid} : t_{\mathsf{Gen}} = \ell{-}1\}$ be the set of bids that are issued in slot $\ell{-}1$. Hence, the input for $\mathcal{U}$'s execution in slot ℓ is $(\mathsf{Pool}^{\ell-1}, \mathsf{chain}^{\ell-1}_{\mathcal{U}}, P^{\ell-1})$.

Next, with the $\mathsf{UpdatePool}$ algorithm given in Algorithm 1, the user obtains her updated bidpool for generating blocks in slot ℓ, *i.e.*, $\mathsf{Pool}^{\ell}_{\mathcal{U}} \leftarrow \mathsf{UpdatePool}(\mathsf{Pool}^{\ell-1}, \mathsf{chain}^{\ell-1}_{\mathcal{U}}, P^{\ell-1})$. We require that the bidpool to be valid as in Definition 11, *i.e.*, $1 \leftarrow \mathsf{VerifyPool}(\mathsf{Pool}^{\ell}_{\mathcal{U}}, \mathsf{chain}^{\ell-1}_{\mathcal{U}})$. By performing the PoBA scheme as a prover, the user obtains a set of valid block candidates, *i.e.*, $\mathcal{U}$ samples bid sets with $\mathsf{BIDs} \leftarrow \mathsf{SampleBIDs}(\mathsf{Pool}_{\mathcal{U}}^{\ell}, \mathsf{N})$ and generates candidates with $\mathsf{bk}_{\mathcal{U}}^{\ell} \leftarrow \mathsf{Solve}(\mathsf{chain}_{\mathcal{U}}^{\ell-1}, \mathsf{BIDs})$. Denote the set of candidates as $BK_{\mathcal{U}} \triangleq \{\mathsf{bk}_{i,\mathcal{U}}^{\ell}\}_{i\in\mathbb{N}}$. Our protocol requires honest users only diffuse their highest-scored block candidate and the corresponding blockchain, *i.e.*, $\mathcal{U}$ diffuses the block $\mathsf{bk}_{\mathcal{U}}^{\ell}$ that satisfies $s_{\mathrm{bk}}(\mathsf{bk}_{\mathcal{U}}^{\ell}) = \max_{\mathsf{bk}\in BK_{\mathcal{U}}} \mathsf{bk}$, and the corresponding blockchain $\mathsf{chain}_{\mathcal{U}}^{\ell-1}$.

In addition to chain diffusion, users also receive blockchains from others. Here, we consider the process of the honest $\mathcal{U}$ updating her local block-tree $\mathsf{tree}^{\ell-1} \triangleq (V^{\ell-1}, E^{\ell-1})$ when receiving a blockchain (including her own) $\mathsf{chain}_{\mathcal{U}^*}^{t'}$. We use $\mathcal{U}^*$ for unspecified users and t' for the slot index to be verified. The user $\mathcal{U}$ first performs as the verifier in PoBA with $(b, \cdot) \leftarrow \mathsf{PoBA.Eval}(\mathsf{chain}_{\mathcal{U}^*}^{t'}, \mathsf{N})$. If $b = 0$, $\mathcal{U}$ discards the blockchain. Otherwise, she compares the incoming blockchain with her local block-tree and adds missing blocks with hash links to the tree. Concretely, when $b = 1$, we have $t' = \ell$. Then, we can rewrite the incoming blockchain with $\mathsf{chain}_{\mathcal{U}^*}^{\ell} = \mathsf{bk}^{G}||\mathsf{bk}_{\mathcal{U}^*}^{1}||\ldots||\mathsf{bk}_{\mathcal{U}^*}^{\ell-1}||\mathsf{bk}_{\mathcal{U}^*}^{\ell}$. Note that users accept *blockchains*. Hence, we only need to consider a sub-chain $\mathsf{bk}_{\mathcal{U}^*}^{t}||\ldots||\mathsf{bk}_{\mathcal{U}^*}^{\ell} \subseteq \mathsf{chain}_{\mathcal{U}^*}^{\ell}$ such that $\mathsf{bk}_{\mathcal{U}^*}^{t-1} \in \mathsf{tree}^{\ell-1}$ and $\mathsf{bk}_{\mathcal{U}^*}^{t} \notin \mathsf{tree}^{\ell-1}$. Therefore, denote the updated block-tree as $\mathsf{tree}_{\mathcal{U}}^{\ell} = (V_{\mathcal{U}}^{\ell}, E_{\mathcal{U}}^{\ell})$, we have $V_{\mathcal{U}}^{\ell} = V^{\ell-1} \cup \{\mathsf{bk}_{\mathcal{U}^*}^{t}, \ldots, \mathsf{bk}_{\mathcal{U}^*}^{\ell}\}$ and $E_{\mathcal{U}}^{\ell} = E^{\ell-1} \cup \{(\mathsf{bk}_{\mathcal{U}^*}^{t-1}, \mathsf{bk}_{\mathcal{U}^*}^{t}), \ldots (\mathsf{bk}_{\mathcal{U}^*}^{\ell-1}, \mathsf{bk}_{\mathcal{U}^*}^{\ell})\}$. We summarize the update process of block-trees with an algorithm: $\mathsf{tree}' \leftarrow \mathsf{UpdateTree}(\mathsf{tree}, \mathsf{chain}_{\mathcal{U}^*}^{t'})$. Note that we use tree and tree' instead of specifying slot indices ($\ell-1$ and ℓ) because users may receive more than one blockchain candidate within one slot. The algorithm is specified in Algorithm 2.

Finally, the user is responsible for reporting her confirmed blockchain to the protocol with respect to a parameter k (to be estimated in Section 6), *i.e.*, given the user's updated block-tree $\mathsf{tree}_{\mathcal{U}}^{\ell}$, the user outputs $\mathsf{chain}_{\mathcal{U}}^{\ell}$ according to Equation 4 such that $\mathsf{chain}_{\mathcal{U}}^{\ell\lceil k}$ is the confirmed part of the blockchain.

Our PoBA-Based Blockchain Protocol $\Pi^{n,\delta,k,s_{\mathrm{bk}},\mathsf{acc}}$

Let $\mathcal{U}$ be an honest user among the n users executing the protocol $\Pi^{n,\delta,k,s_{\mathrm{bk}},\mathsf{acc}}$ in time slot $\ell \geq 1$. Here, δ is the known network delay, k is a parameter for blockchain confirmation, $s_{\mathrm{bk}}(\cdot)$ is the general scoring function for blocks, and $\mathsf{acc}(\cdot)$ is the accumulating parameter function for branch scores. Given the algorithms $\mathsf{UpdatePool}$, $\mathsf{VerifyPool}$, $\mathsf{UpdateTree}$ and the PoBA scheme PoBA, the user $\mathcal{U}$ takes as input $(\mathsf{Pool}^{\ell-1}, \mathsf{tree}^{\ell-1}, P^{\ell-1})$ from the previous slot $\ell-1$.

– **Bidpool Maintenance:** At the beginning of slot ℓ, $\mathcal{U}$ updates the bidpool with $\mathsf{Pool}_{\mathcal{U}}^{\ell} \leftarrow \mathsf{UpdatePool}(\mathsf{Pool}^{\ell-1}, \mathsf{chain}_{\mathcal{U}}^{\ell-1}, P^{\ell-1}$ such that $1 \leftarrow \mathsf{VerifyPool}(\mathsf{Pool}_{\mathcal{U}}^{\ell}, \mathsf{chain}_{\mathcal{U}}^{\ell-1})$ where $\mathsf{chain}_{\mathcal{U}}^{\ell-1} \subseteq \mathsf{tree}^{\ell-1}$ is selected according to Equation 4;

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

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

- **Block Generation:** $\mathcal{U}$ generates block candidates with the PoBA scheme, *i.e.*, $\mathsf{BIDs} \leftarrow \mathsf{PoBA.SampleBIDs}(\mathsf{Pool}_{\mathcal{U}}^{\ell}, \mathsf{N}), \mathsf{chain}_{\mathcal{U}}^{\ell-1} || \mathsf{bk}_{\mathcal{U}}^{\ell} \leftarrow \mathsf{PoBA.Solve}(\mathsf{chain}_{\mathcal{U}}^{\ell-1}, \mathsf{BIDs})$ where N is the size bound of input bid sets;
- **Block-Tree Update:** $\mathcal{U}$ initializes an intermediate variable $\mathsf{tree} \overset{\Delta}{=} \mathsf{tree}^{\ell-1}$. Whenever $\mathcal{U}$ receives a blockchain candidate $\mathsf{chain}_{\mathcal{U}^*}^{t'}$, she runs $\mathsf{tree}' \leftarrow \mathsf{UpdateTree}(\mathsf{tree}, \mathsf{chain}_{\mathcal{U}^*}^{t'})$ and updates her temporary variable with $\mathsf{tree} \leftarrow \mathsf{tree}'$. Finally, the user obtains $\mathsf{tree}_{\mathcal{U}}^{\ell}$ after updating her local block-tree with all blockchain candidates received in slot ℓ;
- **Bids Collection:** $\mathcal{U}$ collects a set of bids issued in slot ℓ as $P^{\ell} \overset{\Delta}{=} \{\mathsf{bid} : t_{\mathsf{Gen}} = \ell - 1\}$ such that all $\mathsf{bid} \in P$ have valid signatures;
- **Ledger Reporting:** Upon queried by the protocol Π, $\mathcal{U}$ outputs her blockchain $\mathsf{chain}_{\mathcal{U}}^{\ell} \subseteq \mathsf{tree}_{\mathcal{U}}^{\ell}$ that satisfies Equation 4, and regard $\mathsf{chain}_{\mathcal{U}}^{\ell\lceil k}$ as the confirmed part of the blockchain.

Discussion: incentive model. The security proofs of this paper are based on practical *assumptions* in the sense of implementation. However, we do *NOT* analyze the reason behind these assumptions based on rational analysis as shown in [2, 3, 11]. Given the richness of the area, we only discuss the intuition of the incentive model. 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 protocol derives from where users can tweak transactions to benefit themselves, *e.g.*, assigning higher buy price for their sell bids or lower sell price for their buy bids. However, prioritizing their own bids in the solving algorithm will potentially sacrifice the block scores. Hence, the 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.

However, problems arise when considering explicit incentives, *i.e.*, rewards to block generators. We argue that a well-chosen scoring function in PoBA prevents adversarial users from attacking the underlying P2PET and from disturbing consensus in the network (as shown in Section 6). However, the computation in PoBA is unfair, *i.e.*, adversaries can dominate the block generation without dominating computing power. The explicit incentive intensifies unfairness.

6 Security Analysis

This section proves the security of our protocol $\Pi^{n,\delta,k}$ with respect to ledger properties, *i.e.*, persistence and liveness [12]. We first provide the definition of persistence by adopting the slightly refined version from [9]. For liveness, existing definitions require that if an honest user receives a *transaction*, then, the transaction will eventually be output by all honest users in their ledger, *i.e.*, blockchain. However, as shown in previous sections, transactions are not diffused solely in our protocol but are released within blocks. Moreover, each user maintains a local block-tree to keep tracking *blocks* of the protocol. Hence, we define block-liveness instead of the original liveness to fit our design.

Definition 15 (Persistence and Block-Liveness). *Denote blockchain as* chain *and block-tree as* tree.

- *Persistence: For any two honest users with blockchains* $\mathsf{chain}_1^{\ell_1}, \mathsf{chain}_2^{\ell_2}$ *at time slot* $\ell_1, \ell_2 \geq 1$*, respectively. Without loss of generality, let* $\ell_1 \leq \ell_2$*. Persistence with parameter* $k \in \mathbb{N}$ *indicates that* $\mathsf{chain}_1^{\ell_1}$*, should be a prefix of* $\mathsf{chain}_2^{\ell_2}$ *after removing the rightmost* k *blocks;*
- *Block-liveness: For any honest user with* chain^{ℓ} *in time slot* $\ell \geq 1$*, block-liveness states that for any* $t \in [\ell]$*,* chain^{t} *is extended by at exact one block.*

Remark 2 (Relaxation in Liveness). As discussed above, we cannot define liveness for transactions in our protocol due to the change in the data structure. There are two options: one is to define liveness for blocks as in Definition 15, which can be considered as a relaxation of the original liveness property, and in fact, our protocol satisfies block-liveness by design (Theorem 2). Whereas, the other option is to define liveness for bids, which can be meaningless in real life. This is because our main purpose is to require honest users to find good assignments (higher-scored blocks) according to the scoring function derived from the underlying P2PET system instead of enforcing them to include every bid.

Persistence. In order to prove persistence, we first make an additional assumption on the adversary. Recall the rushing adversary in our execution model who can learn all block (blockchain) candidates generated, *i.e.*, the adversary holds the master-tree of the protocol. We assume that this adversary sends the highest-scored *blockchain* of each time slot to at least one honest user.

Assumption 1 *Let* mtree^ℓ *be the master-tree in time slot* ℓ*, and let* $\mathcal{A}$ *be the rushing adversary who holds* mtree^ℓ*. Let* $\mathsf{chain}^\ell \subseteq \mathsf{mtree}^\ell$ *be the highest-scored branch that satisfies:*

$$S_{\mathsf{chain}^\ell} = \max_{\mathsf{branch} \subseteq \mathsf{mtree}^\ell} S_{\mathsf{branch}}. \tag{5}$$

We assume that at least one honest user among the n *users who participate in the protocol receives* chain^ℓ *by the end of slot* ℓ*.*

Next, we consider honest users' local block-tree dynamics. It takes two steps to achieve persistence: (1) The highest-scored branch of each time slot should be eventually known to all honest users; (2) Highest-scored branches of different time slots should have a long enough common prefix.

Disclosing highest-scored branches. If a block or branch is known to all honest users, we say it is disclosed, *i.e.*, given a block bk (or a branch branch), for any honest user with block-tree tree, it holds $\mathsf{bk} \in \mathsf{tree}$ (or $\mathsf{branch} \subseteq \mathsf{tree}$). Remark that given the δ-bounded communication network, a block candidate generated by an honest user is always disclosed after δ time slots. Generally, we define d-disclosure for blocks and branches.

Definition 16 (d-Disclosure). *Let* bk^t *be a valid block generated in time slot* $t \geq 1$*, the block is d-disclosed if for any honest user with block-tree* tree^ℓ *in slot* ℓ *such that* $\ell \geq t+d$*, then,* $\mathsf{bk}^t \in \mathsf{tree}^\ell$*. We say a branch is d-closed if the rightmost block on the branch is d-disclosed.*

The following lemma indicates that if a branch is selected as the highest-scored branch by any user, the branch will eventually be disclosed.

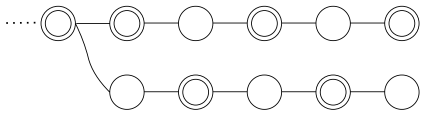
Lemma 1. *Let* mtree^t *be the master-tree in time slot* $t \geq 1$*. Assuming the network is* δ*-synchronous, if a branch* $\mathsf{branch}^t \subseteq \mathsf{mtree}^t$ *is selected as the highest-scored branch according to Equation 5, the branch is at most* $(\delta+1)$*-disclosed.*

Proof. Denote the branch with $\mathsf{branch}^t = \mathsf{bk}^G || \dots || \mathsf{bk}^t$, we first consider the situation where bk^t is generated by an honest user, then, it is δ-disclosed by δ-bounded network setting. Hence, branch is also δ-disclosed. Otherwise, the block is first received by the rushing adversary $\mathcal{A}$ in slot t. Because $\mathsf{branch}^t \subseteq \mathsf{mtree}^t$ is the highest-scored branch according to Equation 5, by Assumption 1, an honest user will receive the branch in slot t. Denote the user's local block-tree with tree^t. Since tree^t is a sub-graph of mtree^t, branch is the highest-scored branch in tree^t. Hence, the honest user will generate a block atop branch in the following slot $t+1$, which is also δ-disclosed. Therefore, the branch is at most $\delta+1$-disclosed.

Directly from Lemma 1, we have the following proposition for all disclosed highest-scored branches in the master-tree.

Proposition 1. *Assuming the network is δ-slot synchronous, let* mtree^t *be the master-tree in time slot t in which* branch^t *is selected as the highest-scored branch according to Equation 5. All blocks on branches in* $\{\mathsf{branch}^t\}_{t\in[\ell-(\delta+1)]}$ *is disclosed for any $\ell > \delta + 1$.*

Common prefix among selected branches. Proposition 1 only indicates that the highest-scored branch of each time slot will eventually be known to all honest users. However, even in the master-tree (*i.e.*, 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 time slot. Such a substitution causes the blockchain to be unstable and prevents honest users from agreeing on the same *chain*. We consider two situations: (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) If the change of branch selection happens frequently, the block history cannot be settled (Figure 1a); (2) If the selected branches of different slots have too many distinct blocks, the block history can get reset (Figure 1b). In either case, invalid bids and transactions in the unconfirmed blocks can disturb the underlying P2PET market.


(a) The selected chain swings over conjunctive time slots so that the blocks during these slots cannot be settled.

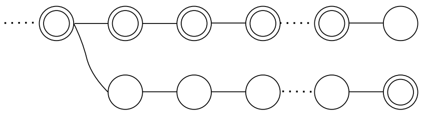

(b) A branch substitutes the selected chain after it gets selected for multiple slots so that the history gets reset.

Fig. 1: Intuition of Chain Instability: In the successful attack, the adversary forces the nodes to keep changing the chain between two cases.

In order to tackle this problem, we first define divergence between branches and branch viability. The definitions originate from [17], we refine them with respect to the score of branches. Moreover, since we focus on the highest-scored branches in each time slot, and they are disclosed as shown in Proposition 1, we will consider the master-tree in the following analysis for simplicity.

Definition 17 (Divergence and Viability). *Let* mtree^ℓ *be the master-tree of time slot $\ell \geq 1$. For any $t \in [\ell]$, given any two branches* $\mathsf{branch}_1^t, \mathsf{branch}_2^t \subseteq \mathsf{mtree}^t$*, the divergence of* branch_1^t *and* branch_2^t *is given by:*

$$\mathrm{div}(\mathsf{branch}_1^t, \mathsf{branch}_2^t) = |S_{\mathsf{branch}_1^t} - S_{\mathsf{branch}_2^t}|.$$

Moreover, let chain^t *be the highest-scored branch selected from* mtree^t *according to Equation 5. A branch* $\mathsf{branch}_*^t \neq \mathsf{chain}^t$ *is viable, if* $\mathrm{div}(\mathsf{branch}_*^t, \mathsf{chain}^t) \leq \mathsf{smax}{-}\mathsf{smin}$ *where* $[\mathsf{smin}, \mathsf{smax}]$ *is the range of block scores given by* $s_{bk}(\cdot)$.

A branch in slot t is viable if the divergence between the branch and the highest-scored branch can be covered by a single block. That is, the viable branch could be extended with a block with higher score and substitute the selected chain. We formally define this situation as chain substitution with a parameter $\tau \geq 1$. For convenience, we first introduce the injection operation for branches. Let $\mathsf{branch}_1 = \mathsf{bk}^G || \dots || \mathsf{bk}^t || \mathsf{bk}_1^{t+1} \dots || \mathsf{bk}_1^{\ell_1}$ be a branch of slot $\ell_1 \geq 1$, and let $\mathsf{branch}_2 = \mathsf{bk}^G || \dots || \mathsf{bk}^t || \mathsf{bk}_2^{t+1} \dots || \mathsf{bk}_2^{\ell_2}$ be a branch of slot $\ell_2 \geq 2$ where $t \in [\min(\ell_1, \ell_2)]$. Then, the intersection of branch_1 and branch_2 is denoted by $\mathsf{branch}_1 \cap \mathsf{branch}_2 = \mathsf{bk}^G || \dots || \mathsf{bk}^t$.

Definition 18 (τ-Chain Substitution). *Let* mtree^ℓ *be the master-tree of time slot* $\ell{>}\tau$*. For any* $t \in [\ell{-}\tau]$*, let* chain^i *and* chain^{i+1} *be the highest-scored branches in slot* i *and* $i{+}1$ *where* $i \in [t{-}1, t{-}2{+}\tau]$ *such that* $\mathsf{chain}^{i+1} = \mathsf{chain}^i || \mathsf{bk}_\mathsf{c}^{i+1}$*. Let* $\mathsf{bk}_\mathsf{c}^{t+\tau}$ *be highest-scored block that extends* $\mathsf{chain}^{t-1+\tau}$ *in slot* $t{+}\tau$*. Denote the new branches* $\mathsf{branch}_\mathsf{c} \overset{\Delta}{=} \mathsf{chain}^{t-1+\tau} || \mathsf{bk}_\mathsf{c}^{t+\tau}$*. A* τ*-chain substitution occurs if there exists a branch such that* $\mathsf{branch}^{t-1+\tau} \cap \mathsf{chain}^{t-1+\tau} = \mathsf{chain}^{t-1}$*, and is extended by block* $\mathsf{bk}_\mathsf{b}^{t+\tau}$*, denote the new branch* $\mathsf{branch}_\mathsf{b} \overset{\Delta}{=} \mathsf{branch}^{t-1+\tau} || \mathsf{bk}_\mathsf{b}^{t+\tau}$*, such that:*

$$S_{\mathsf{branch}_\mathsf{b}} = \max_{\mathsf{branch}^{t+\tau} \subseteq \mathsf{mtree}^{t+\tau}} S_{\mathsf{branch}^{t+\tau}}.$$

A chain can only be substituted by a branch if the branch is viable, and the blocks extending them satisfies $s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{b}^{t+\tau}) - s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{c}^{t+\tau}) \geq S_{\mathsf{chain}^{t-1+\tau}} - S_{\mathsf{branch}^{t-1+\tau}}$, *i.e.*, $S_{\mathsf{branch}_\mathsf{b}} \geq S_{\mathsf{branch}_\mathsf{c}}$. The parameter $\tau \geq 1$ indicates that a chain is substituted after being selected for τ conjunctive slots. Assuming *arbitrary* distribution $\mathcal{D}$ for our scoring function $s_{\mathrm{bk}}(\cdot)$, the following lemma shows a loose upper bound for the probability of τ-chain substitution.

Lemma 2. *Let* mtree^ℓ *be the master-tree of time slot* $\ell{>}\tau$*. For any* $t \in [\ell{-}\tau]$*, given any score distribution* $\mathcal{D}$*, there exists an accumulating function* $\mathsf{acc}(\cdot)$ *(Equation 3) such that the probability of* τ*-chain substitution occurring is* $\mathsf{O}(c^{-\tau})$ *where* c *is a constant value given by* $\mathsf{acc}(\cdot)$*.*

Proof. We start from the easy case where $\tau = 1$. By Definition 18, $\mathsf{chain}^t = \mathsf{chain}^{t-1} || \mathsf{bk}_\mathsf{c}^t$ and $\mathsf{branch}_\mathsf{c} = \mathsf{chain}^t || \mathsf{bk}_\mathsf{c}^{t+1}$. Consider a viable branch that satisfies $\mathsf{branch}^t \cap \mathsf{chain}^t = \mathsf{chain}^{t-1}$ and is extended by a block $\mathsf{bk}_\mathsf{b}^{t+1}$ such that $\mathsf{branch}_\mathsf{b} = \mathsf{branch}^t || \mathsf{bk}_\mathsf{b}^{t+1}$ substitutes $\mathsf{branch}_\mathsf{c}$. Then, by Equation 3, we have:

$$\begin{aligned} s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{c}^t) &\geq s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{b}^t), \\ \frac{\mathsf{acc}(t)}{\mathsf{acc}(t{+}1)} s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{c}^t) + \cdot s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{c}^{t+1}) &\leq \frac{\mathsf{acc}(t)}{\mathsf{acc}(t{+}1)} s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{b}^t) + s_{\mathrm{bk}}(\mathsf{bk}_\mathsf{b}^{t+1}). \end{aligned} \tag{6}$$

For simplicity, we omit the subscript in the scoring function with $s(\cdot)$ as we only consider block scores. We also rewrite $\frac{\mathsf{acc}(t)}{\mathsf{acc}(t+1)}$ with $c(0,1)$. Then, the probability of $\mathsf{branch_c}$ substituted by $\mathsf{branch_b}$, denoted by $\Pr[\tau = 1, \mathsf{branch_b}]$, equals to joint probability of events in Equation 6. That is, we need to estimate the following.

$$\Pr[s(\mathsf{bk}^t_\mathsf{c}) - s(\mathsf{bk}^t_\mathsf{b}) \geq 0 \wedge c(0,1) \cdot \big(s(\mathsf{bk}^t_\mathsf{c}) - s(\mathsf{bk}^t_\mathsf{b})\big) + \big(s(\mathsf{bk}^{t+1}_\mathsf{c}) - s(\mathsf{bk}^{t+1}_\mathsf{b})\big) \leq 0] \tag{7}$$

Now, we denote the random variables representing the scores of the tuple $(\mathsf{bk}^t_\mathsf{c}, \mathsf{bk}^t_\mathsf{b}, \mathsf{bk}^{t+1}_\mathsf{c}, \mathsf{bk}^{t+1}_\mathsf{b})$ with $(X^t_\mathsf{c}, X^t_\mathsf{b}, X^{t+1}_\mathsf{c}, X^{t+1}_\mathsf{b})$. Furthermore, we use two random variables Y^t and Y^{t+1} to represent the subtraction of scores. That is,

$$Y^t = X^t_\mathsf{c} - X^t_\mathsf{b} = s(\mathsf{bk}^t_\mathsf{c}) - s(\mathsf{bk}^t_\mathsf{b}),$$
$$Y^{t+1} = X^{t+1}_\mathsf{c} - X^{t+1}_\mathsf{b} = s(\mathsf{bk}^{t+1}_\mathsf{c}) - s(\mathsf{bk}^{t+1}_\mathsf{b}).$$

Following in our universal sampler model, $(X^t_\mathsf{c}, X^t_\mathsf{b}, X^{t+1}_\mathsf{c}, X^{t+1}_\mathsf{b})$ 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* [13] 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 $\mathcal{D}_Y$. We can rewrite Equation 7 in the form of random variables:

$$\Pr[(Y^t \geq 0) \wedge c(0,1) \cdot Y^t + Y^{t+1} \leq 0)]. \tag{8}$$

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 is independent of Y^{t+1}, we have:

$$\begin{aligned}\text{Equation 8} &= \int_0^r \Pr[Y^t = y] \cdot \Pr[Y^{t+1} \leq -c(0,1)y] \mathrm{d}y \\ &= \int_0^r \int_{-r}^{-c(0,1)y} f_Y(y) f_Y(x) \mathrm{d}x\mathrm{d}y. \end{aligned} \tag{9}$$

Here, we consider the upper-bound of Equation 9 by scaling up $f_Y(\cdot)$ with two coefficients $c_1, c_2 > 0$ as follows.

$$f_Y(y) \begin{cases} \leq c_1 \cdot e^{-c_2 \cdot y^2}, & \text{if } y \in [-r, r], \\ = 0, & \text{otherwise.} \end{cases} \tag{10}$$

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:

$$\text{Equation 9} \leq \int_0^\infty \int_{-\infty}^{-c(0,1)y} e^{-y^2} \cdot e^{-x^2} \mathrm{d}x\mathrm{d}y = \frac{{c_1}^2}{2c_2} \cdot \tan^{-1}\left(\frac{1}{c(0,1)}\right) \leq \frac{{c_1}^2}{2c_2 \cdot c(0,1)}. \tag{11}$$

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

Now, we consider the probability of 1-chain substitution, denoted by $\Pr[\tau = 1]$. Let q be the number of viable branches of slot t. Recall our universal sampler,

Q is the upper bounded of the total number of queries (regardless of honest or not) made in each time slot, *i.e.*, $q \leq Q$ as shown in Section 4.3 (universal sampler functionality). Then, we have:

$$\Pr[\tau = 1] \leq 1 - \left(1 - \frac{c_1^2}{2c_2 \cdot c(0,1)}\right)^Q .$$

Next, **chain substitution with $\tau > 1$** follows the same methodology of $\tau = 1$. Consider a branch that satisfies $\mathsf{branch}^{t-1+\tau} \cap \mathsf{chain}^{t-1+\tau} = \mathsf{chain}^{t-1}$. Denote the distinct blocks on $\mathsf{branch}^{t-1+\tau}$ with $\mathsf{bk}_\mathsf{b}^t, \ldots, \mathsf{bk}_\mathsf{b}^{t-1}+\tau$. Comparing them with the blocks on $\mathsf{chain}^{t-1+\tau}$, and comparing the new blocks $\mathsf{bk}_\mathsf{b}^{t+\tau}$ in $\mathsf{branch}_\mathsf{b}$ and $\mathsf{bk}_\mathsf{c}^{t+\tau}$ in $\mathsf{branch}_\mathsf{c}$, by rewriting Equation 8, the probability of $\mathsf{branch}_\mathsf{c}$ substituted by $\mathsf{branch}_\mathsf{b}$, denoted by $\Pr[\tau > 1, \mathsf{branch}_\mathsf{b}]$, equals to:

$$\Pr\left[\left(\bigwedge_{i\in[\tau-1]} \left(\sum_{j=0}^{i} \frac{\mathsf{acc}(t+j)}{\mathsf{acc}(t+i)} \cdot Y^{t+j} \geq 0\right)\right) \wedge \left(\sum_{i=0}^{\tau} \frac{\mathsf{acc}(t+i)}{\mathsf{acc}(t+\tau)} \cdot Y^i \leq 0\right)\right], \quad (12)$$

where $Y^{t+i} = X_\mathsf{c}^{t+i} - X_\mathsf{b}^{t+i}$ for any $i \in \{0, \ldots, \tau\}$ are independent and distributed identically with $\mathcal{D}_Y$ on $[\mathsf{smin}-\mathsf{smax}, \mathsf{smax}-\mathsf{smin}]$. For simplicity, we rewrite $\frac{\mathsf{acc}(j)}{\mathsf{acc}(i)}$ with $c(j,i)$, and let $c(i,i) \overset{\Delta}{=} 1$. Hence, Equation 12 equals to:

$$= \int_0^r \cdots \int_{r_{\tau-1}}^r \mathit{\Pi}_{i=0}^{\tau-1} \Pr[Y^{t+i} = y_i] \cdot \Pr\left[Y^{t+\tau} \leq -\sum_{i=0}^{\tau-1} c(i,\tau) \cdot y_i\right] \mathrm{d}y_0 \cdots \mathrm{d}y_{\tau-1}$$

$$= \int_0^r \cdots \int_{r_{\tau-1}}^r \int_{-r}^{-\sum_{i=0}^{\tau-1} c(i,\tau)\cdot y_i} \mathit{\Pi}_{i=0}^{\tau-1} f_Y(y_i) \cdot f_Y(x) \mathrm{d}x \mathrm{d}y_0 \cdots \mathrm{d}y_{\tau-1}. \quad (13)$$

We denote the lower bound of random variable Y^{t+i} for any $i \in [\tau-1]$ as r_i. Hence $r_i = -\sum_{j=0}^{i-1} c(j,i) \cdot y_j$. Here, we use a small trick to scale Equation 13. Note that r_i is not necessarily larger than 0. We scale y_i down to $-r$ for all $i \in [\tau-1]$. Then, the upper bound of $Y^{t+\tau}$ satisfies $Y^{t+\tau} \leq -c(0,\tau)y_0 + r \cdot \sum_{i=1}^{\tau-1} c(i,\tau)$. By $\int_{-r}^{r} f_Y(y_i)\mathrm{d}y_i \leq 1$ for any $i \in [\tau-1]$, we have:

$$\text{Equation 13} \leq \int_0^r \int_{-r}^{-c(0,\tau)y_0 + r\cdot\sum_{i=1}^{\tau-1} c(i,\tau)} f_Y(y_0) f_Y(x) \mathrm{d}x \mathrm{d}y_0. \quad (14)$$

Finally, by setting $y' \overset{\Delta}{=} y_0 - \frac{r}{c(0,\tau)} \cdot \sum_{i=1}^{\tau-1} c(i,\tau)$ and the bound of $f_Y(\cdot)$ given in Equation 10:

$$\text{Equation 14} \leq \int_{-\infty}^{\infty} \int_{-\infty}^{-c(0,\tau)y'} e^{-y'^2} \cdot e^{-x^2} \mathrm{d}x \mathrm{d}y' = \frac{{c_1}^2}{c_2} \cdot \tan^{-1}\left(\frac{1}{c(0,\tau)}\right) \leq \frac{c_1^2}{c_2} \cdot c(0,\tau). \quad (15)$$

That is, $\Pr[\tau > 1, \mathsf{branch}_\mathsf{b}] \leq {c_1}^2/c_2 \cdot c(0,\tau)\}$, for any $c_1, c_2 \geq 0$ and $c_1^2/c_2 \geq \pi^{-1}$ where $c(0,\tau) = \frac{\mathsf{acc}(t)}{\mathsf{acc}(t+\tau)}$. Then, similar to $\tau = 1$, we compute the probability of

$\tau \geq 1$-chain substitution, denoted by $\Pr[\tau > 1]$, as follows. Let q be the number of viable branches in slot t. Then, by our universal sampler, $q \leq Q$, where Q is the upper bound of the total number of queries (regardless of honest or not) made in each time slot. Hence,

$$\Pr[\tau > 1] \leq 1 - \left(1 - \frac{c_1^2}{c_2 \cdot c(0, \tau)}\right)^Q .$$

Combining the discussion above, we consider a constant value $c > \max\{1, \frac{c_1^2}{c_2}\}$, and let $\mathsf{acc}(t) = c^t$. 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 $Q \cdot c^{-\tau}$. Note that we only use the upper bound of total queries to the universal sampler instead of honest queries. Finally, we can conclude that τ-chain substitution occurs with probability $\mathsf{O}(c^{-\tau})$.

Therefore, we have the following theorem on persistence.

Theorem 1 (Persistence). *Assuming at least one honest user, the protocol $\Pi^{n,\delta,k,s_{bk},\mathsf{acc}}$ among the n users, it holds that the protocol parameterized with $k \geq \delta+1$ satisfies persistence (Definition 15) with probability at least $1 - \Omega(c^{-k+\delta})$.*

Proof. Suppose persistence with parameter $k \geq \delta + 1$ is violated. It follows that, for honest users in two different time slots $\ell_1 \leq \ell_2$ holding $\mathsf{chain}_1^{\ell_1}$ and $\mathsf{chain}_2^{\ell_2}$, $\mathsf{chain}_1^{\ell_1}\lceil^k$ is not the prefix of $\mathsf{chain}_2^{\ell_2}$. Hence, there exists blocks on $\mathsf{chain}_1^{\ell_1}\lceil^k$ not on $\mathsf{chain}_2 \neq \emptyset$. Denote the first distinct block with $\mathsf{bk}_1^{\ell_1 - k'}$, then $k' \geq k$.

By Proposition 1, in slot ℓ_2, highest-scored branches of any slot before $\ell_2 - (\delta + 1)$ are disclosed to all honest users. Hence, the block-tree tree_2^{ℓ} of the user who holds $\mathsf{chain}_2^{\ell_2}$ contains all highest-scored branches before $\ell_2 - (\delta + 1)$. Then, $\mathsf{chain}_1^{\ell_1}\lceil^{\delta+1} \in \mathsf{tree}_2^{\ell}$. The chain substitution must happen in the slot after $\ell{-}\delta{+}1$. Without loss of generality, we consider the situation that for all $i \in \{\ell_1{-}k', \ldots, \ell_1{-}(\delta{+}1)\}$, $\mathsf{chain}_1^i = \mathsf{chain}_1^{i-1}||\mathsf{bk}^i$ for all $i \in \{\ell_1{-}k', \ldots, \ell_1{-}(\delta{+}1)\}$, *i.e.*, blocks on chain^{ℓ_1} are selected conjunctively for $k'{-}\delta$ slots from $\ell{-}k'$ to $\ell{-}(\delta{+}1)$. Therefore, by Lemma 2, the probability of this $(k'{-}\delta)$-chain substitution occurs with probability of $\mathsf{O}(c^{-k'+\delta})$ which is less then $\mathsf{O}(c^{-k+\delta})$. Finally, we conclude that the probability of persistence with $k \geq \delta{+}1$ is at least $1{-}\Omega(c^{-k+\delta})$.

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

Theorem 2 (Block-liveness). *Assuming at least one honest user executes the protocol $\Pi^{n,\delta,k,s_{bk},\mathsf{acc}}$ among the n users, it holds that the protocol satisfies block-liveness (Definition 15) unconditionally.*

The proof is straightforward. Assuming the one honest user is unaware of any other blocks, she can trivially extend her block-tree by generating blocks locally with the bidpool and selecting the blockchain accordingly.

7 Final Remarks

The starting point of our protocol is to implement the ledger of the P2PET, where the underlying problem of matching buy and sell bids, can be used in conjunction with a blockchain data structure. Despite the fact that users need to be certificated to participate in the system, we adopt a more general execution: permissionless with static corruptions and δ-synchronous communication network. Moreover, we assume the existence of a globally synchronized clock, which, at first glance, is a constrained setting. However, it can easily be implemented by the secure hardware provided by the P2PET system.

One component of our construction is the scoring function for blocks, *i.e.*, s_{bk}, and the BAP-based "proof-of-X" scheme (PoBA), implementing a novel consensus protocol. The purpose of the scoring function is to provide a "notion of optimal" to the system. Among all the blocks available in each time slot, it chooses the "most optimal" choice to extend a (redefined) blockchain data structure. At the same time, it is used as accounting for the auction market underpinning the system. Instantiated with a concrete function, the adversary, once given the description for the scoring function, could adapt and get an advantage in constructing the next block. Thus, its adaptability, in fact, helps the optimality of the overall protocol. We advocate that the study of more concrete constructions of s_{bk} is of independent interest and out of the scope of this work.

In our analysis, the PoBA scheme is replaced by a *universal sampler* which samples blocks and corresponding scores from score space in each time slot. This, in fact, simplifies our analysis without the loss of the whole motivation of our construction. We remark that, although it is an advanced random oracle, it is practical [16], since it can be easily used in practice with a random function.

Considering the universal sampler in our protocol execution setting, we showed that our protocol has persistence and block-liveness when at least one user is honest. We recall that *block-liveness* is a variant of the standard security liveness property. This variant is necessary and meaningful in our setting, given that in every time slot *all* honest participants issue candidate blocks. Thus, the property is that one block among them is always chosen from all candidates. Needless to say, by fulfilling the block-liveness property, we also obtain the regular liveness property, thereby constructing a fully secure system.

Finally, we remark that we did not thoroughly investigate potential incentive frameworks for our proposed system in the presence of a rational adversary. This topic seem to be out of the scope of the current work despite its importance. In particular, the study of the overall behavior of the system in the presence of such adversary. We leave this topic for future works.

References

1. Akhras, R., El-Hajj, W., Majdalani, M., Hajj, H.M., Jabr, R.A., Shaban, K.B.: 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. pp. 1672–1678. IEEE (2020). https://doi.org/10.1109/IWCMC48107.2020.9148345, `https://doi.org/10.1109/IWCMC48107.2020.9148345`
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.) 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. Lecture Notes in Computer Science, vol. 10821, pp. 34–65. Springer (2018). https://doi.org/10.1007/978-3-319-78375-8“˙2, `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.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 3–32. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9“˙1, `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.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10991, pp. 789–819. Springer (2018). https://doi.org/10.1007/978-3-319-96884-1“˙26, `https://doi.org/10.1007/978-3-319-96884-1\_26`
5. Blocki, J., Zhou, H.: Designing proof of human-work puzzles for cryptocurrency and beyond. In: Hirt, M., Smith, A.D. (eds.) Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9986, pp. 517–546 (2016). https://doi.org/10.1007/978-3-662-53644-5“˙20, `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 Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA. pp. 136–145. IEEE Computer Society (2001). https://doi.org/10.1109/SFCS.2001.959888, `https://doi.org/10.1109/SFCS.2001.959888`
7. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Lecture Notes in Computer Science, vol. 740, pp. 139–147. Springer (1992). https://doi.org/10.1007/3-540-48071-4“˙10, `https://doi.org/10.1007/3-540-48071-4\_10`
8. 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, `https://doi.org/10.1007/BF01581197`
9. Fitzi, M., Kiayias, A., Panagiotakos, G., Russell, A.: Ofelimos: Combinatorial optimization via proof-of-useful-work \\ A provably secure blockchain protocol. In: Advances in Cryptology - CRYPTO 2022. Lecture Notes in Computer Science, Springer (2022)

10. Foundation, C.: Cardano Hub. `https://www.cardano.org/` (2023), [Online; accessed 26-May-2023]
11. Garay, J.A., Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Rational protocol design: Cryptography against incentive-driven adversaries. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA. pp. 648–657. IEEE Computer Society (2013). https://doi.org/10.1109/FOCS.2013.75, `https://doi.org/10.1109/FOCS.2013.75`
12. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) 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. Lecture Notes in Computer Science, vol. 9057, pp. 281–310. Springer (2015). https://doi.org/10.1007/978-3-662-46803-6“˙10, `https://doi.org/10.1007/978-3-662-46803-6\_10`
13. George E.P. Box, G.C.T.: Bayesian Assessment of Assumptions 1. Effect of Non-Normality on Inferences about a Population Mean with Generalizations, chap. 3, pp. 149–202. John Wiley and Sons, Ltd (1992). https://doi.org/https://doi.org/10.1002/9781118033197.ch3, `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118033197.ch3`
14. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988). https://doi.org/10.1137/0217017, `https://doi.org/10.1137/0217017`
15. 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.) Computer Aided Systems Theory - EUROCAST 2019 - 17th International Conference, Las Palmas de Gran Canaria, Spain, February 17-22, 2019, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 12013, pp. 507–514. Springer (2019). https://doi.org/10.1007/978-3-030-45093-9“˙61, `https://doi.org/10.1007/978-3-030-45093-9\_61`
16. 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.) 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. Lecture Notes in Computer Science, vol. 10032, pp. 715–744 (2016). https://doi.org/10.1007/978-3-662-53890-6“˙24, `https://doi.org/10.1007/978-3-662-53890-6\_24`
17. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 357–388. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7“˙12, `https://doi.org/10.1007/978-3-319-63688-7\_12`
18. Kirli, D., Couraud, B., Robu, V., Salgado-Bravo, M., Norbu, S., Andoni, M., Antonopoulos, I., Negrete-Pincetic, M., Flynn, D., Kiprakis, A.: Smart contracts in energy systems: A systematic review of fundamental approaches and implementations. Renewable and Sustainable Energy Reviews **158**, 112013 (2022)
19. Luu, L., Chu, D., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi,

S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 254–269. ACM (2016). https://doi.org/10.1145/2976749.2978309, https://doi.org/10.1145/2976749.2978309

20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), http://bitcoin.org/bitcoin.pdf
21. 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)
22. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: Nielsen, J.B., Rijmen, V. (eds.) 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. Lecture Notes in Computer Science, vol. 10821, pp. 3–33. Springer (2018). https://doi.org/10.1007/978-3-319-78375-8"¨1, https://doi.org/10.1007/978-3-319-78375-8_1
23. Robertson, H.: 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"
24. Wood, G.: Ethereum yellow paper (2014)