Bid-Matching Problem and Score-Based Consensus for Peer-to-Peer Energy Trading

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

[1] Department of Mathematical and Computing Sciences, School of Computing, Tokyo Institute of Technology. Tokyo-to Meguro-ku Ookayama 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 (P2P) energy trading systems (ETS) grows alongside the development of house renewable energy generation. A P2P/ETS enables its peers to trade energy freely as in a double auction market. It requires a ledger to record peers' trading history. A typical approach is relying on a decentralized ledger, *e.g.*, blockchain, with smart contract capabilities, unavoidably incurring high costs. Therefore, motivated to build a smart contract-free system, this work proposes a novel blockchain and consensus design utilizing the double auction characteristics of P2P/ETS. Concretely, we first revisit the blockchain data structure so that it can reflect auction bids. Next, we introduce a novel mining mechanism utilizing a bid-matching problem (BMP), which requires miners to find the best combination sets of sell/buy bids according to a given scoring function. Hence, the miner who mines the best-scored block can extend the blockchain. The fundamental difference between the BMP-based mining and traditional proof-of-X schemes, *e.g.*, work or stake, is that our protocol selects blocks instead of miners. That is, a higher-scored block has better contents (bids and transactions), thus being preferable to a lower-scored block regardless of whether the miner is honest. Finally, we analyze miners' local chain dynamics and show a bound for the score distribution of the scoring function to prove that the protocol satisfies the key properties of consensus, *i.e.*, persistence and liveness.

Keywords: Peer-to-Peer Energy Trading, Double Auction, Blockchain Mining, Score-Based Consensus.

1 Introduction

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 system to the decentralized bi-direction "smart grid". The new structure requires a peer-to-peer (P2P) energy trading system (ETS) that enables users to trade energy and

exchange data to achieve regional self-sustainability. We abstract P2P/ETS as a double auction market where users can create bids to sell/buy energy among themselves. Therefore, a ledger is in demand to record bids and trading transactions. Whereas in the decentralized setting, blockchain protocols [16] have been proven to fulfill the ledger properties [7], *i.e.*, persistence and liveness.

The existing works [1, 11] on blockchain-based P2P/ETS rely on smart contracts capabilities [15], *e.g.*, typically the one provided by Ethereum [21]. Unfortunately, the smart contract-equipped protocols suffer from high maintenance fees [18], where users must incur severe costs while submitting smart contract-based transactions to the network. Moreover, the widely-used underlying mining mechanism, *i.e.*, the proof-of-work scheme [16], is intensifying the global energy crisis. Therefore, this work focuses on an unexplored alternative to circumvent these prohibitive costs by introducing P2P/ETS's double auction characteristics into the protocol design. Our protocol will work for P2PETS and can be generalized to be compatible with double auction markets, which is an advantage even for smart contract-equipped protocols with low transaction fees, *i.e.*, Cardano/Ouroboros [6].

Abstract P2P/ETS. In order to better reason the technical challenges of a blockchain protocol tailored to P2P/ETS, we abstract the system concerning its use case and settings. Concretely, the users in a P2P/ETS are equipped with advanced metering devices that measure and record their energy generation and trades. The users are often geographically close to each other in a tight-knit community, *e.g.*, a village. However, this is not necessary because the system can be extended hierarchically, *i.e.*, multiple local systems can form a higher level of ETS. A crucial assumption is a good network connection, *i.e.*, with less delay, among users in the same system. We overlook transmission loss since such analysis is out of the scope of this work.

In the abstract P2P/ETS, users continuously present sell/buy bids for each other. The key idea of establishing trades is to require users to find the most optimal set of matched bids (transactions) within a constrained period, *i.e.*, a time slot. Hence, the system can enforce desired matching policy by defining optimum, *e.g.*, maximizing trading volume or preserving variants of fairness [2, 12]. Later, we push forward and formalize this idea as the bid-matching problem (BMP) with a given scoring function.

1.2 Our Contributions

To the best of our knowledge, we are the first to merge the setting of users operating in a double auction market and the blockchain protocol. The following Table 1 lists some blockchain-based consensus and the typical proof-of-X schemes, including work, stake, and weight-based work. We note that more protocols, *e.g.*, [4, 5, 17], are not shown in the table.

Our contributions are threefold. **First**, we add a bid layer to the blockchain data structure so that the protocol can handle auction bids from the P2P/ETS

Table 1: Some Noticeable Proof-of-X-Based Protocols.

Protocol	Mining Mechanism	Chain Selection
Bitcoin Backbone [7]	Proof-of-work	Longest-chain
Ouroboros [6, 14]	Proof-of-stake	Longest-chain
Weight Nakamoto [13]	Weight-based proof-of-work	Weight-based
This work	Score-based BMP	Highest-score

or double auction markets. Under the new data structure, we present a variant version of the ledger properties introduced in [7]. **Second**, we propose a BMP-based mining mechanism that utilizes an optimization problem to match sell/buy bids for transactions according to a publicly known scoring function. Here, the scoring function serves as a reflection of desired market dynamics. The block is easy to generate. However, it should be hard to find a high-scored one. Our mining mechanism (BMP and the scoring function) guarantees that a higher-scored block implies a more desirable set of tradings for the underlying auction market. Our design is novel in the sense that this is the first time a block's quality is decided by its content rather than the resource outside the protocol, *e.g.*, work or stake in the traditional proof-of-X schemes. Moreover, Existing blockchain analysis [6,7,17] often separates their market participants and miners. In contrast, we regard them as sub-procedures of a single entity so that the miner may benefit from its market participant (Details can be found in Section 5.2). We argue that this setting is more close to practice. **Third**, inspired by the forkable string formalism in [14], we consider a *tree structure* for users' view of blocks because the BMP-based mining can generate blocks easily. Moreover, we push forward the idea of the scoring function to a score accumulator that evaluates the score of branches in the tree. The chain is selected with the "highest-score" rule, *i.e.*, in a time slot, the branch with the highest accumulated score is adopted by all honest users as their blockchain. We analyze honest users' local chain (tree) dynamics and prove that, with carefully chosen protocol parameters, the ledger properties are achievable for arbitrary score distribution of the scoring function.

1.3 Organization

We organize the main contents of the paper as follows. First, the preparation to present the protocol starts in Section 2, where we review the notations and protocol settings. Section 3 introduces our tailored "bid-transaction-block-chain" data structure and the corresponding modification to the ledger properties. Next, in Section 4, we formalize the BMP-based mining mechanism and provide the scoring oracle model by discussing a concrete scoring function. Section 5 introduces the concrete protocol description. Finally, we present the security analysis on users' local chain dynamics in Section 6.

2 Notation and the Execution Model

Throughout this paper, we use λ for the security parameter and $||$ for concatenation. For a set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ denotes that x is randomly and uniformly sampled from $\mathcal{X}$; Whereas for an algorithm Alg, $x \leftarrow \mathsf{Alg}$ denotes that x is assigned the

output of an algorithm Alg on fresh randomness. We denote the set $\{0, \ldots, \ell - 1\}$ with $[\ell]$.

For completeness, we employ digital signatures with Existential Unforgeability under Adaptive Chosen Message Attacks ($\mathsf{EUF\text{-}CMA}$) [10]. In general, a digital signature scheme is a tuple of three PPT algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ such that:

- $\mathsf{KeyGen}(1^\lambda)$ takes in a security parameter and outputs a verification key pk and a signing key sk.
- $\mathsf{Sign}_{\mathsf{sk}}(m)$ takes in a signing key sk and a message m, outputting a signature σ on message m under signing key sk.
- $\mathsf{Verify}_{\mathsf{pk}}(m, \sigma)$ takes in a verification key pk, a message m and a signature σ, outputting 1 if the signature is valid and 0 otherwise.

In addition to the signature definition, let Hash denote a collision-free hash function. Next, we describe our protocol's execution model and introduce the ledger properties.

Execution model. We adopt the standard Interactive Turing Machines (ITM) approach to model protocol execution [3]. A protocol refers to an algorithm for a set of nodes (ITMs) to interact with each other. The execution of the protocol Π is directed by the environment $\mathcal{Z}(1^\lambda)$, where λ is the security parameter. The environment controls the spawn and corruptions of nodes in each protocol instance. We consider that all corrupt nodes are controlled by an adversary $\mathcal{A}$ who can read inputs and set outputs for these nodes. Moreover, we assume the existence of a globally synchronized clock $\mathcal{T}$, secured with the signature scheme introduced above, such that every node can access directly, *i.e.*, without the online adversary knowing, to obtain the current time slot.

- **Round and time slot.** The protocol execution proceeds in rounds, being the smallest unit of time of interest. At the beginning of each round, honest nodes receive inputs from the environment $\mathcal{Z}$ and send outputs to $\mathcal{Z}$ at the end of the round. The round number is non-decreasing. Honest nodes' views of the round number may be slightly different from each other. We further divide execution into time slots, each consisting of multiple rounds. A time slot is associated with a block updating the blockchain.
- **Permissioned setting and static corruption.** Since our motivation and application are constrained to P2P/ETS, this work considers the *permissioned* setting, requiring that $\mathcal{Z}$ spawn all nodes before the protocol execution and inform the identities of all nodes spawned to the honest ones. Furthermore, we assume the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ in our protocol respect static corruption. Hence, $\mathcal{Z}$ is not allowed to halt nodes or issue corrupt instruction to an honest node after the beginning of the execution, *i.e.*, $\mathcal{Z}$ spawns corrupted nodes directly.
- **Communication network.** We adapt Canetti [3]'s synchronous communication description to our time slot setting. Each message sent by an honest node is guaranteed to arrive at most the next slot after the delay of δ rounds,

which is *known* by the nodes. That is, all the messages sent to a node at slot $r - \delta$ are received before the node sends any slot-r related messages.

3 Basic Definitions

Here, we start to explain our design. The first step is to define the data structure on which our protocol relies, *i.e.*, a "bid-transaction-block-chain" structure. Hence, we revisit the ledger properties, *i.e.*, persistence and liveness as discussed in [7] and present a variant of the liveness property for our new data structure.

3.1 Data Structure

Consider a double auction market where users start their operations by creating bids. Within each bid, the user has two options: (1) to sell a quantity of "something" for an initial price (anything upper is acceptable), or (2) to buy a quantity of "something" for a price (anything lower is acceptable). A transaction is a pair of matched buy bid and sell bid, a block contains a set of bids and transactions, and a chain is a linked list of blocks.

In the following, we denote each instance as bid, tx, bk, and chain. For convenience, each instance is associated with an identifier ID, and the identifier is set to be the hash of the instance's contents, *e.g.*, $\mathsf{ID}_{\mathsf{bid}} = \mathsf{Hash}(\mathsf{bid})$ is the identifier of a bid. Within the definitions, the public keys of each user are known. We assume the existence of the globally synchronized clock $\mathcal{T}$, equipped with a public key pair $(\mathsf{pk}_{\mathcal{T}}, \mathsf{sk}_{\mathcal{T}})$, which the users can submit queries $(\sigma_{\mathcal{T}}, t) \leftarrow \mathcal{T}(m)$, such that $\sigma_{\mathcal{T}} = \mathsf{Sign}_{\mathsf{sk}_{\mathcal{T}}}(m, t)$. Moreover, as discussed in Section 5.2, we use $\mathcal{U} = (\mathcal{B}, \mathcal{M})$ to denote a user's bidder and miner sub-procedures.

Definition 1 (Bid). *A* bid *issued by a user's bidder with public and secret keys* $(\mathsf{pk}_{\mathcal{B}}, \mathsf{sk}_{\mathcal{B}})$ *consists of the tuple* $(\mathsf{raw}_{\mathsf{bid}}, t, \sigma_{\mathsf{bid}}, \mathsf{pk}_{\mathcal{B}}, \mathsf{misc})$*:*

- $\mathsf{raw}_{\mathsf{bid}} = (\mathsf{action}, q, p)$ *denotes the action, quantity, and price of the bid. In particular, the* action *has two possible values:* $\{\mathsf{buy}, \mathsf{sell}\}$*;*
- t *is the slot number provided by the globally synchronized clock* $\mathcal{T}$*;*
- σ_{bid} *is the signature of the raw bid, that is* $\sigma_{\mathsf{bid}} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathcal{B}}}(\mathsf{raw}_{\mathsf{bid}}, t)$*;*
- misc *contains additional information, e.g., a certificate by a trusted authority attesting the buyer or seller's public key.*

Each user of the protocol keeps a list of all bids seen on the network, in a similar fashion as the *mempool* of blockchain systems (with the difference that in such systems, the mempool keeps transactions). As one should expect, a transaction is the combination of two bids; one for *selling* and the other for *buying* at a combined price that would satisfy both bids.

Definition 2 (Transaction). *Let* bid_1 *and* bid_2 *be two bids such that* $\mathsf{raw}_1 = (\mathsf{sell}, q_1, p_1)$ *and* $\mathsf{raw}_2 = (\mathsf{buy}, q_2, p_2)$*. Therefore, a transaction* tx *consists of* $(\mathsf{ID}_{\mathsf{bid}_1}, \mathsf{ID}_{\mathsf{bid}_2}, q_{\mathsf{tx}}, p_{\mathsf{tx}})$ *where* $q_{\mathsf{tx}}, p_{\mathsf{tx}}$ *denote the matched quantity and price.*

A transaction matches a buy bid and a sell bid. The correctness of matching can be found in Section 4.1. Next, our proposed abstraction is finally described by introducing the definition of block and chain.

Definition 3 (Block). *A block* bk *consists of the tuple* $(\mathsf{raw}_{\mathsf{bk}}, \mathsf{pk}_{\mathcal{M}}, t, \mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}}, \mathsf{score}, h, \sigma_{\mathcal{M}})$*:*

- $\mathsf{raw}_{\mathsf{bk}} = (\mathsf{BIDs}, \mathsf{TXs})$ *is the raw contents of the block, that is the bid list* BIDs *and the transaction list* TXs*;*
- $\mathsf{pk}_{\mathcal{M}}$ *is the public key of the miner* $\mathcal{M}$*, and its signature* $\sigma_{\mathcal{M}} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathcal{M}}}(\mathsf{raw}_{\mathsf{bk}}, t, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{score}, h)$*;*
- t *is the slot number from* $\mathcal{T}$*, and* $\mathsf{pk}_{\mathcal{T}}$ *is the public key. It signs with* $\sigma_{\mathcal{T}} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathcal{T}}}(\mathsf{raw}_{\mathsf{bk}}, t)$*;*
- score *is the score of the block with respect to its content* $\mathsf{raw}_{\mathsf{bk}}$*;*
- $h = \mathsf{Hash}(\mathsf{bk}_{\mathsf{prev}})$ *is the hash of the previous block.*

In order to define the chain itself, as an initial assumption, we assume the existence of the *Genesis Block*, denoted by bk_G, which is publicly known by the users. This initial block contains the public keys of the users, which are allowed to issue block candidates. Therefore, 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). *A chain* chain *is a ordered list of blocks* $(\mathsf{bk}_G, \mathsf{bk}^1, \mathsf{bk}^2, \ldots)$ *such that* bk_G *is the genesis block which contains the public keys of the users of the system.*

3.2 Ledger Properties

Typically in distributed ledgers, each transaction changes the state of the protocol. Thus, it is convenient to introduce a notation to address this framework. Assume the existence of two states q_1 and q_2, let $q_1 \xrightarrow{\mathsf{tx}} q_2$ denote the state change from q_1 to q_2 introduced by the transaction tx. Moreover, let $q \xrightarrow{*} q'$ denote the transition between two states caused by a finite number of transactions. Under our protocol execution setting, we recall the ledger properties as discussed in [7].

- **Persistence.** For any two nodes, say, S_1 and S_2, and any two slots $\ell_1 \leq \ell_2$, if the list of settled transaction of S_1 in ℓ_1 is equal to LOG_1, and the total transactions for S_2 is LOG_2, thus LOG_1 is a prefix of LOG_2;
- **Liveness.** If a transaction tx is available to all nodes at a point when the latest slot for the honest nodes is t, then any node whose clock advances u slots, to the point of t', we will have a ledger such that the state q, for which it holds that $q_0 \xrightarrow{*} q_1 \xrightarrow{\mathsf{tx}} q_2 \xrightarrow{*} q$.

Each node would keep locally a chain chain, which can be seen as a log, *i.e.*, LOG, which keeps a list of transactions of the whole history of the protocol. Therefore, two different logs LOG_1 and LOG_2, such that the latter contains the former, give us that LOG_1 is also known as the "prefix of " LOG_2.

Liveness variant. Furthermore, we adapt the liveness property to our setting, introducing an analogous property with respect to blocks, instead of transactions. Unlike other consensus or leader election mechanisms, we insist that a relaxation in the liveness property is necessary for our setting because any participant can put forth a block with a trivial, *i.e.*, not optimal, combination of bids. Therefore, in each time slot, as many blocks as participants can potentially be available, and the decision of the new block depends on the scoring function (later introduced in detail in Section 4.2, which, internally, can take several features of the set of transactions of the block. Another significant difference is that bids, once broadcasted, will not necessarily be included in the new blocks as transactions, *i.e.*, they may not be combined to another bid to make a transaction, for example. In our protocol, the more suitable feature is the notion that among all the candidate blocks, one will be used to extend the chain. Moreover, transactions are not broadcasted solely in our design but released within the *blocks* that are issued respectively by all the players in each time slot. Therefore, we overload the earlier introduced notation $q \xrightarrow{tx} q'$ for blocks, *i.e.*, $q \xrightarrow{\mathsf{bk}} q'$, denoting the state change for a batch of transactions within bk. For these reasons and without loss of generality, we rewrite the liveness property as follows:

- **Block-liveness**: If a set of blocks $(\mathsf{bk}_0, \mathsf{bk}_1, \mathsf{bk}_2, \mathsf{bk}_3, \ldots, \mathsf{bk}_n)$ is available to all honest nodes at a point when the latest slot for the honest nodes is t, then any node whose clock advances u slots, to the point of t', we will have a ledger such that the state q, for which it holds that $q_0 \xrightarrow{*} q_1 \xrightarrow{\mathsf{bk}_i} q_2 \xrightarrow{*} q$ for some i such that $1 \leq i \leq n$.

4 Bid-Matching Problem and the Scoring Function

The bid-matching-based mining derives from P2P/ETS's double auction characteristic and relies on the data structure introduced in Section 3.1. The users keep a list of bids to form a block of transactions. Users are expected to maximize the matching score with respect to a publicly known scoring function $s : \mathsf{bk} \to \mathbb{R}$. s is a multidimensional function that maps the content of a block, *e.g.*, number of bids/transactions, matched quantity/price, timestamps, etc., to a real number. This section first presents a concise definition of the bid-matching problem (BMP). Next, we describe the scoring function s and provide an oracle model for security analysis. Finally, we briefly discuss the differences between our BMP-based mining and traditional proof-of-X schemes.

4.1 Bid-Matching Problem

Our scheme is based on the optimization problem of matching bids. We define it as the Bid-Matching Problem (BMP).

Definition 5 (BMP). *Given the publicly known scoring function* $s : \mathsf{bk} \to \mathbb{R}$ *and the current blockchain* chain*, a user* $\mathcal{U}$ *solving the BMP, for a bid list* BIDs *it*

keeps with respect to a probabilistic matching algorithm Matching*, outputs a block* $\mathsf{bk} = (\mathsf{raw}_{\mathsf{bk}}, \mathsf{pk}_{\mathcal{U}}, \sigma_{\mathcal{U}}, t, \mathsf{pk}_{\mathcal{T}}, \sigma_{\mathcal{T}}, \mathsf{score}, h)$ *with* $\mathsf{raw}_{\mathsf{bk}} = (\mathsf{BIDs}, \mathsf{TXs})$ *such that the three algorithms* (Matching, Verify, Scoring) *perform as follows:*

- $\mathsf{Matching}(\mathsf{BIDs}; r) \to \mathsf{TXs}$*. On input of* BIDs *and an explicit randomness* r*,* Matching *outputs a transaction list* TXs*, where each* $\mathsf{tx} \in \mathsf{TXs}$ *contains a pair of buy/sell bids listed in* BIDs*;*
- $\mathsf{Verify}(\mathsf{chain}, \mathsf{BIDs}, \mathsf{TXs}) \to \{0, 1\}$*. Any user can run the deterministic algorithm* Verify *and check the correctness of each transaction* tx *in the list* TXs *with respect to the blockchain* chain *and the input bid list* BIDs*;*
- $\mathsf{Scoring}(\mathsf{bk}) \to \mathsf{score}$*. The deterministic algorithm* Scoring *evaluates the scoring function* $s(\mathsf{bk})$*.*

Matching correctness. Notice that the input BIDs can contain residual bids, *i.e.*, the bid's quantity not being fully matched in the current blockchain. Such bids can be taken as input for the Matching algorithm (Details in Section 5.2). Hence the Verify algorithm also needs to check the history embedded in the blockchain. Let $\boldsymbol{H} = \{\mathsf{TXs}_1, \ldots, \mathsf{TXs}_n\}$ be the history transactions embedded in the blockchain chain. For a newly found transaction list TXs in a block candidate, each $\mathsf{tx} = (\mathsf{bid}_{\mathsf{buy}}, \mathsf{bid}_{\mathsf{sell}}, p_{\mathsf{tx}}, q_{\mathsf{tx}}) \in \mathsf{TXs}$ should satisfies:

- $p_{\mathsf{sell}} \leq p_{\mathsf{tx}} \leq p_{\mathsf{buy}}$;
- $\sum_{\mathsf{TXs}_i \in \boldsymbol{H}} \sum_{\mathsf{tx}_{\mathsf{buy}} \in \mathsf{TXs}_i} q_{\mathsf{tx}_{\mathsf{buy}}} + \sum_{\mathsf{tx}_{\mathsf{buy}} \in \mathsf{TXs}} q_{\mathsf{tx}_{\mathsf{buy}}} \leq q_{\mathsf{buy}}$;
- $\sum_{\mathsf{TXs}_i \in \boldsymbol{H}} \sum_{\mathsf{tx}_{\mathsf{sell}} \in \mathsf{TXs}_i} q_{\mathsf{tx}_{\mathsf{sell}}} + \sum_{\mathsf{tx}_{\mathsf{sell}} \in \mathsf{TXs}} q_{\mathsf{tx}_{\mathsf{sell}}} \leq q_{\mathsf{sell}}$.

A transaction is a match between a buy bid and a sell bid. The matched price should be in the range between the sell bid's price and the buy bid's price. Otherwise, the buyer cannot afford the seller's bid. Moreover, since we enable miners to do matchings across time slots, the sum of the matched quantity of an involving buy/sell bid should not surpass the total quantity of the bid.

4.2 The Scoring Function and the Scoring Oracle Model

Now, we proceed to the Scoring algorithm and the scoring function s. Because the BMP can be regarded as an optimization problem on a bipartite graph, its complexity and output distribution depend on the concrete scoring function. Moreover, it relates to the users' behavior considering rational analysis, *e.g.*, by taking typical parameters into account, users may match such bids together for higher scores. Thus, a well-designed scoring function is necessary for the BMP-based mining mechanism.

In order to better reason our model, we explicitly rewrite the scoring function as $s = T \circ s_L$. Here, $s_L : \mathsf{bk} \to \mathbb{R}$ is a multidimensional *linear* function that maps the block generated by the Matching algorithm to a real number. $T : \mathbb{R} \to \mathbb{R}$ is a transformation [20] that transforms $s_L(\cdot)$'s linearity, *e.g.*, by setting $T(x) = e^x$. As shown in Figure 1a, the output score distribution can be changed by the transformation $T(\cdot)$. In the following, we first discuss the potential of the scoring function by showing a concrete example.

A concrete example. A concrete instantiation of the scoring function is tailored to implement desired properties of the underlying trading market and the consensus (blockchain) protocol. In terms of the trading market, for example, the system can be designed to privilege blocks with particular properties, such as the highest number of matched transactions, or the highest transacted value among all the matched pairs of sell and buy bids, *i.e.*, aggregated value of all transactions, or even, consider time features of the bids, *i.e.*, oldest bids would have higher score, etc. Whereas, as we will discuss in detail in Section 6, the convergence, *e.g.*, linearity and convexity, of the scoring function provided by the transformation T, can change users' local chain dynamics and the probability of them achieving consensus over time.

As a starting point, we give a toy example by setting the market goal as maximizing the total matched price (unit price times quantity) with a linear transformation $T(x) = x$. On a given transaction list TXs, we write the scoring function concerning the goal as follows:

$$s(\mathsf{TXs}) = T \circ s_L(\mathsf{TXs}) = T(\sum_{\mathsf{tx} \in \mathsf{TXs}} p_{\mathsf{tx}} \cdot q_{\mathsf{tx}}) = \sum_{\mathsf{tx} \in \mathsf{TXs}} p_{\mathsf{tx}} \cdot q_{\mathsf{tx}}. \tag{1}$$

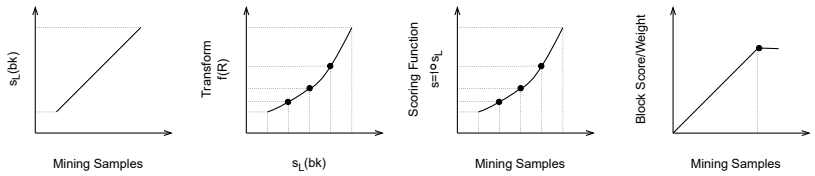
Notice that this goal is very different from the goal of a traditional double auction market, *e.g.*, the stock market, we use it to demonstrate our scoring function design's versatility. Since $p_{\mathsf{sell}} \leq p_{\mathsf{tx}} \leq p_{\mathsf{buy}}$, a miner would prefer to choose $p_{\mathsf{tx}} = p_{\mathsf{buy}}$ to maximize the score. We can control this by setting $p_{\mathsf{tx}} = \epsilon \cdot p_{\mathsf{sell}} + (1-\epsilon) p_{\mathsf{buy}}$ with an explicit $\epsilon \in [0, 1]$ to adjust preference for buyers or sellers.

With the scoring function in Equation 1, the BMP can be reduced to a generalized assignment problem [19], which is proven to be NP-hard. However, being NP-hard is insufficient to analyze the distribution of output scores if users use an arbitrary $\mathsf{Matching}$ algorithm. Therefore, we offer users a generic stochastic local search algorithm as $\mathsf{Matching}$ for finding the solution that maximizes the scoring function. A potentially "more clever" algorithm may outperform honest participants who stick with the protocol. However, we argue that the offered algorithm is general enough to close this gap.

Therefore, we make an restriction that each instance of the protocol performed by the users $(\mathcal{U}_1, \mathcal{U}_2, \dots)$ is defined by the tuples $(\mathsf{Matching}(\cdot; r_1), \mathsf{Verify}, \mathsf{Scoring})$, $(\mathsf{Matching}(\cdot; r_2), \mathsf{Verify}, \mathsf{Scoring})$, $\dots$, respectively. Here, r_i is the randomness of $\mathcal{U}_i$. Following the analysis given in [7], *i.e.*, computing power is uniformly distributed among users, whereas the adversary who corrupts a fraction of users controls the fraction of computing power, we assume r_i follows the *uniform distribution*. Hence, the output of s_L follows the uniform distribution since each participant follows the same searching process with only different randomness. The difference between our model and the previous analysis is that, with the transformation in the scoring function, our design can lift the final score outputs from the uniform distribution to a desirable distribution $\mathcal{D}$. Finally, we summarize a "scoring oracle" model, which is able to model both the concrete example and more general scoring function cases for the security analysis.

Scoring oracle model. The scoring oracle $\mathcal{O}$ is accessible by any $\mathcal{U}_i \in \mathsf{Usr}$, via queries in which the user sends its own bid list BIDs_i and the randomness r_i. All the users have direct communication with it, *i.e.*, the communication between $\mathcal{U}_i$ and the oracle cannot be delayed by the adversary. This is a natural setting since the oracle captures the capability of the players to internally and locally select the bids and issue the block relying only on its randomness and the fixed matching algorithm. Next, we describe its internal routines.

The score sampling routine. The score is generated by assuming a *scoring space* S and a *random sampling algorithm* $\mathsf{Sampling}$ such that in slot t, $(\mathsf{score}_i^t, \mathsf{bk}_i^t) \leftarrow \mathsf{Sampling}(r_i^t, r_{\mathcal{O}}^t, s)$ for the user's randomness r_i^t, the oracle's internal randomness $r_{\mathcal{O}}^t$, and the scoring function s. The algorithm has the single property of randomly sampling $\mathsf{score}_i^t \in \mathsf{S}$ that follows a given distribution $\mathcal{D}$ according to the scoring function. The $\mathsf{Sampling}$ algorithm also returns the corresponding block. Note that this setting is enough for our purposes, given that with a suitably large enough set S, the blocks can be strictly ordered by the score with high probability.



(a) This work. (b) In [8, 13].

Fig. 1: Intuition of Scoring Function and Scoring Oracle.

The oracle. More concretely, given a scoring function $s = T \circ s_L$, for each slot t, $\mathcal{O}$ picks a random value $r_{\mathcal{O}}^t$, and for every time slot, it keeps lists of performed queries $\mathbb{L}^t = \{(\mathcal{U}_1, \mathsf{score}_1^t, \mathsf{bk}_1^t), (\mathcal{U}_2, \mathsf{score}_2^t, \mathsf{bk}_2^t), \ldots, (\mathcal{U}_n, \mathsf{score}_n^t, \mathsf{bk}_n^t)\}$ for n users, and upon receiving a query (BIDs_i, r_i) from $\mathcal{U}_i$.

Scoring Oracle $\mathcal{O}$

For every slot t, the oracle keeps n-sized lists $\mathbb{L}^t = \{(\cdot,\cdot,\cdot), \ldots, (\cdot,\cdot,\cdot)\}$ and picks a uniformly random value $r_{\mathcal{O}}^t$. **On a query** $(\mathsf{BIDs}_i^t, r_i^t)$ **from** $\mathcal{U}_i$**:** If $\mathcal{U}_i \in \mathbb{L}_t$ such that there is a tuple $(\mathcal{U}_i, \mathsf{score}_i^t, \mathsf{bk}_i^t)$, then return $(\mathsf{score}_i^t, \mathsf{bk}_i^t)$ to $\mathcal{U}_i$. Otherwise

- Sample $(\mathsf{score}_i^t, \mathsf{bk}_i^t) \leftarrow \mathsf{Sampling}(r_i^t, r_{\mathcal{O}}^t, s)$ where $s(\mathsf{bk}_i^t) = \mathsf{score}_i^t$;
- Add $(\mathcal{U}_i, \mathsf{score}_i^t, \mathsf{bk}_i^t)$ to $\mathbb{L}^t$;
- Return the pair $(\mathsf{score}_i^t, \mathsf{bk}_i^t)$ to $\mathcal{U}_i$.

The scoring oracle model captures the restriction mentioned above by sampling on the uniformly distributed users' randomness. It outputs the score and the corresponding block following the distribution given by the scoring function.

The output distribution. Therefore, we model the output score of our oracle as a continuous random variable X follows a specific distribution $\mathcal{D}$ over score range $\mathsf{S} = [\mathsf{smin}, \mathsf{smax}]$. Here, $\mathcal{D}$ is given by the transformation T on the uniformly distributed random variables output by s_L. We denote the probability density function and the distribution function of $\mathcal{D}$ as $f(\cdot)$ and $F(\cdot)$, respectively. Hence, $F(x) = \Pr[X \leq x] = \int_{\mathsf{smin}}^{x} f(t)dt$. In any time slot, regardless of the distribution, for n users including t adversaries, one of the adversaries' blocks has the highest score with probability t/n. Thus, suppose we assume the honest majority, the probability of the adversary winning a time slot is less than $1/2$.

4.3 Mining Fairness and the Content-Related Mining

One may notice that the BMP-based mining mechanism lacks a crucial property of traditional proof-of-X, which is mining fairness. Usually, in proof-of-X schemes, mining fairness requires that the probability of a user's block being selected to extend the blockchain is comparable to the fraction of resources it uses in the mining process. Let Res be the required resource in a proof-of-X scheme, *e.g.*, *hash power* for proof-of-work and *stake* for proof-of-stake. We have $\Pr[\mathcal{U}$ extends the blockchain$] \approx \mathcal{U}$'s Res/Network's Res. Otherwise, the ratio of blocks in the blockchain that are created by the adversary cannot be bounded by its resource, thus being a violation of the chain quality property defined in [7]. Chain quality is crucial because the adversary can control the transactions in its blocks, harming liveness by not including typical transactions.

In contrast, the BMP-based mining in practice does not rule out "clever" users (honest or not) who can take advantage, *i.e.*, finding a higher-scored block with less computing power used in $\mathsf{Matching}$. Later in Remark 1 of Section 5, we even demonstrate a strategy to boost block score without much computation. This is because of the fundamental difference between our design and the traditional proof-of-X schemes, *i.e.*, the BMP-based mining selects blocks instead of miners. A higher-scored block is always preferable to the system because the block's *content* is sophisticatedly chosen according to the scoring function, and the well-designed scoring function enforces a desirable matching policy for the system. Any violation of this policy sacrifices the score of the user (adversary)'s block. Notice that our process is also different from the weight-based proof-of-X frameworks [8, 13] (Illustrated in Figure 1b), because the weight in these works evaluates the quality of proof-of-X instead of the block's content. However, currently, the incentive model is unclear because it cannot handle the situation where *all of* the users start to put low-effort transactions into their blocks so that the overall quality of blocks falls. We list this as our future work.

5 Protocol Description

In this section, we present the whole picture of our protocol including how users handle bids and their local blockchain with the existence of a globally synchronized clock. Intuitively, the protocol is executed by n users such that each one

maintains a local *tree (forest)*, in which the root is a unique "genesis" block bk_G containing the users' public keys. We enable users to store a tree because blocks are easy to generate in the BMP-based mechanism, which requires extra efforts to achieve consensus on a *chain*. In a time slot, each user generates block candidates that are issued via the BMP mining. Then, the user updates its local tree with the new candidates. Finally, it selects the chain according to the "highest-score" rule, *i.e.*, adopting the highest-scored branch in the tree with respect to the current slot number.

5.1 Local Tree Structure

Since our protocol enables users to store tree structures, we first define the combination view of all users as the "master-tree". Hence, each user's view is a sub-tree (sub-forest) of the master-tree. Given the genesis block as the root, a master-tree mtree^t for time slot t is formed from all the valid blocks generated by users (honest or not) from genesis to the end of slot t. Blocks of the same slot would share the same height. A user may only see a part of the master-tree if it is not aware of other blocks generated. We denote the sub-tree of user $\mathcal{U}$, for the set of users Usr, in slot t as $\mathsf{tree}^t_{\mathcal{U}}$, which contains at least all honest blocks up to the slot t. We present the formal definition as follows.

Definition 6 (Master-Tree). *Given a hash function* Hash*, let the genesis block in time slot* 0 *be* bk_G*. For each slot* $t \in \{1, \ldots, \ell - 1\}$*, we denote valid block candidates generated within slot* t *as* $\{\mathsf{bk}^t_{\mathcal{U}}\}_{\mathcal{U}\in\mathsf{Usr}}$*. All* $\mathsf{bk}^1_{\mathcal{U}}$ *are connected to* bk_G *by* Hash*; for any* $t > 1$ *and participant* $\mathcal{U}$*,* $\mathsf{bk}^t_{\mathcal{U}}$ *is connected to one and only one* $\mathsf{bk}^{t-1}_{\mathcal{U}'}$*. A master-tree for this* ℓ *time slots is denoted by* $\mathsf{mtree} = (V, E)$*, where* $V = \bigcup_{t\in[\ell]} \{\mathsf{bk}^t_{\mathcal{U}}\}_{\mathcal{U}\in\mathsf{Usr}}$ *and* $E = \bigcup_{t\in[\ell]]\setminus\{0\}} \{(\mathsf{bk}^{t-1}_{\mathcal{U}'}, \mathsf{bk}^t_{\mathcal{U}})\}$ *where* $(\mathsf{bk}^{t-1}_{\mathcal{U}'}, \mathsf{bk}^t_{\mathcal{U}})$ *denotes the connection between blocks.*

Definition 7 (User-Tree). *Let* $\mathsf{mtree}^\ell = (V, E)$ *be a master-tree of time slot* ℓ*, i.e., at the end of slot* ℓ*. A user-tree of an honest user* $\mathcal{HU} \in \mathsf{HUsr}$ *denoted by* $\mathsf{tree}_{\mathcal{HU}}$ *contains all blocks that the user witnesses by the end of slot* ℓ*. Consider a sub-graph* $\mathsf{htree} = (V^*, E^*)$ *where* $V^* = \bigcup_{t\in[\ell+1],\mathcal{HU}\in\mathsf{HUsr}} \{\mathsf{bk}^t_{\mathcal{HU}}\}$*. The user-tree is a sub-graph of* mtree *and a super-graph of* htree*.*

The outlined definitions are, in spirit, similar to the approach based on the forkable string introduced by [14]. Later, we present a full analysis of the chain dynamics in Section 6. Our tree view approach is fundamentally more suitable to our construction because, differently to [14], in each time slot, potentially each participant issues a block candidate. We remark that each candidate block will extend at most one branch of the master-tree. In comparison, regular proof-of-X-based blockchain protocol decides only a single player to issue the new block,*i.e.*, only one block is generated for each slot. Thus, there is no notion of block candidates in the same sense as our protocol. Next, we recall the following definitions in graph theory for explaining our highest-score rule.

Definition 8 (Branch, height, length). *A path in tree $G = (V, E)$ originating at the root $r \in V$ is called a branch* branch*. Later, we also* branch *for a chain of blocks; The height of a node, denoted with* $\text{height}(v)$ *where $v \in V$, is defined as the edge number on the longest path to a leaf from that node; A branch's length, denoted with* $\text{length}(\mathsf{branch})$*, equals to the edge number on the branch. The height of the tree* $\text{height}(G)$ *is defined as the height of the root or the length of the longest branch.*

Hence, we overload the scoring function for branches as the score accumulator $s^a_{acc} : \mathsf{branch} \to \mathbb{R}$ with an accumulating parameter function $a : \text{height} \to \mathbb{R}$.

Definition 9 (Score Accumulator). *Assume a scoring function $s : \mathsf{bk} \to \mathbb{R}$ and a parameter function $a : \text{height} \to \mathbb{R}$. Given a user-tree* tree *of height h in a time slot, the accumulated score of a branch* $\mathsf{branch} \subseteq \mathsf{tree}$ *is defined as:*

$$s^a_{acc}(\mathsf{branch}) = \sum_{\mathsf{bk} \in \mathsf{branch}, i=0}^{h} a(i) \cdot s(\mathsf{bk}).$$

The function s_{acc} evaluates the score of a branch where each block on the branch is given a parameter according to its height. The function $a(\cdot)$ adds flexibility to the score accumulator, thus the protocol can assign blocks at the typical height with different values. For example, we can give old blocks on the chain higher values to achieve better chain stability (analyzed in Section 6.1).

5.2 Workflow Overview

In order to demonstrate the protocol design with respect to users' workflow, we first separate each user into two sub-procedures: bidder and miner. Briefly, a user keeps a local tree of blocks (user-tree) and a mempool containing all the currently available bids. In a time slot t defined by the globally synchronized clock $\mathcal{T}$ (as cited in Section 2), the user's bidder sub-procedure is expected to create bids within the slot for selling or buying according to its energy demands and broadcast the bids to the network mempool.

Meanwhile, the miner sub-procedure collects bids from the mempool and checks conflicts concerning its current selected blockchain. Here, we enable miners to collect "residual" bids in the bid list BIDs for the $\mathsf{Matching}$ algorithm. Concretely, a residual bid is a bid that has been included in the blockchain with unmatched quantities. For example, given a bid bid^* and all the transactions that embeds the bid: $\{\mathsf{tx}_i\} = \{(\mathsf{bid}^*, \mathsf{bid}_i, q_i, p_i)\} \in \mathsf{chain}^{t-1}$ such that $q_{\mathsf{bid}^*} > \sum_i q_i$. A miner can collect bid^*'s residual as bid' where $q_{\mathsf{bid}'} = q_{\mathsf{bid}^*} - \sum_i q_i$. Such a setting enables users to better fulfill their demands and increases (energy) market's trading efficiency.

Hence, with the collected bid list, the miner runs the BMP mining algorithms to generate a block candidate that, in its view, maximizes the scoring function. It broadcasts its block candidate and receives others candidates. Then, the user updates the user-tree and chooses the highest-scored branch in its blockchain as

given by the score accumulator. The user also updates the mempool accordingly so that no conflicted bid exists. Thus, the users are ready for the next time slot.

The bidder is a participant in the trading market, and the miner is a participant in the blockchain protocol. Previous works [6, 7, 17] make an explicit separation between these two types of participants. However, in this work, we regard them as the sub-procedures of a single entity, *i.e.*, a user, so that the bidder may issue bids on the fly to give its miner advantages (explained in Remark 1). We argue that such a setting is more close to practice.

Remark 1 (Bidders' trick). Let a user's bid list collected from the mempool be BIDs. Without loss of generality, we assume no conflicts between BIDs and the user's current selected chain. Let the transaction list found by the user's miner sub-procedure be TXs. Suppose there exists a not matched (nm) bid in the bid list, $\mathsf{bid}_{\mathsf{nm}} \in \mathsf{BIDs}$, *i.e.*, for any $\mathsf{tx} \in \mathsf{TXs}$, $\mathsf{bid}_{\mathsf{nm}} \notin \mathsf{tx}$. Hence, the user's bidder sub-procedure can create a $\mathsf{bid}_{\mathsf{otf}}$ on the fly (otf) so that a new $\mathsf{tx}_{\mathsf{otf}} = (\mathsf{bid}_{\mathsf{nm}}, \mathsf{bid}_{\mathsf{otf}}, q_{\mathsf{tx}_{\mathsf{otf}}}, p_{\mathsf{tx}_{\mathsf{otf}}})$ can be matched accordingly, and $s(\mathsf{TXs}||\mathsf{tx}_{\mathsf{otf}}) \geq s(\mathsf{TXs})$. Therefore, the user gains mining advantages by creating a new bid instead of performing extra computations in the $\mathsf{Matching}$ algorithm.

Such a trick is often considered a violation of mining fairness mentioned in Section 4.3. However, in our protocol, we do not rule out this trick by two reasons: (1) the user who creates $\mathsf{bid}_{\mathsf{otf}}$ must fulfill the bid, *i.e.*, the user must provide energy for a sell bid or pay for the energy for a buy bid; (2) the $\mathsf{bid}_{\mathsf{otf}}$ and the corresponding $\mathsf{tx}_{\mathsf{otf}}$ enhance the system efficiency according to the scoring function, *e.g.*, boosting the trading volume if we consider the concrete function given in Equation 1.

5.3 Concrete Protocol Description

Assume the protocol Π is run by n participants. Each user $\mathcal{U}$ locally keeps a blockchain $(\mathsf{bk}_G, \mathsf{bk}^1, \mathsf{bk}^2, \ldots, \mathsf{bk}^\ell)$, and a list of $\mathsf{BIDs}_{\mathcal{U}}$. On every new time slot t started by the synchronized clock, the participant receives a new set of bids, and a collection of new blockchains. The protocol also receives the parameter ρ (to be estimated in Section 6) for confirmation of blocks which is used to output the "confirmed blocks". Concretely it is used to prune the locally kept blockchain of the blocks with slot number at most $\ell - \rho$.

The protocol participant performs the following procedures.

Scoring-Based Protocol Π^ρ

- **Bid List Update:** At any point of the time slot t, the bid is received and added to the locally kept list of bids BIDs by bidders and miners;
- **Blockchain Update:** Whenever an alternative blockchain $(\mathsf{bk}_G, \mathsf{bk}^{1\prime}, \mathsf{bk}^{2\prime}, \ldots, \mathsf{bk}^{t\prime})$ with $w > \ell$ is presented, it replaces its local copy and updates the bid list accordingly, *i.e.*, by removing from the local list, the bids presented in the new blockchain;

– **Blockchain Extension:** On the beginning of the time slot, the user $\mathcal{U}$ issues its candidate block by executing $\mathsf{Matching}(\mathsf{BIDs}; r_{\mathcal{U}}) \to \mathsf{bk}^*_{\mathcal{U}}$, and broadcasts $\mathsf{bk}^*_{\mathcal{U}}$ to the network. Upon receiving the candidate blocks $\mathsf{bk} \neq \mathsf{bk}_{\mathcal{U}}$ from the others participants, and verifies each score by sorting $s^c_{acc}(\mathsf{BIDs}_{ji}, \mathsf{bk}_j) \to \mathsf{score}_j$ for all j. Finally it sets $\mathsf{bk}_{\ell+1} \leftarrow \mathsf{bk}^*_{max}$, such that bk^*_{max} is the block with the highest score s_j among all blocks bk_j.
– **Ledger Reporting:** Upon queried, output the $(\mathsf{bk}_G, \mathsf{bk}^1, \mathsf{bk}^2, \ldots, \mathsf{bk}^{\ell'})$, such that $\mathsf{bk}^{\ell'}$ is the block whose time stamp is $\ell - \rho$.

Later, in our analysis, we substitute the calls to function s^c_{acc} in the above description to calls to the random score oracle $\mathcal{O}$ in order to evaluate the score for the blocks.

In the following section, we analyze the users' local chain (tree) dynamics and show how honest users can achieve persistence and block-liveness by adopting the highest-scored branch as their locally kept blockchain.

6 Local Chain Dynamics

In order to achieve persistence, the highest-scored branch should be *known* to all honest users and should *not* deviate too much in conjunctive time slots. Whereas, block-liveness can be directly derived from our protocol setting because all honest users will collect at least one block in a time slot into their user-tree.

6.1 Achieving Persistence

Considering the existence of an adversarial user, the main challenges of achieving persistence are: (1) Branch disclosure: The adversary cannot hide the highest-scored branches from a part of the honest users; (2) Chain stability: The adversary cannot force a swing among branches by extending the non-highest-scored branches with higher-scored blocks.

Branch disclosure. For the first challenge, we remark that given our network setting, in particular the δ-bounded block propagation delay, a block created by an honest user is always disclosed at most within δ slots. Generally, we define *block disclosure* with a parameter k with respect to the master tree mtree^ℓ of time slot ℓ and an honest user $\mathcal{HU} \in \mathsf{HUsr}$'s local trees $\mathsf{tree}^t_{\mathcal{HU}}$.

Definition 10 (k-Disclosure). *Let $\mathsf{mtree}^\ell = (V, E)$ be a master-tree of time slot ℓ, let $V = \bigcup_{t\in[\ell+1]} \{\mathsf{bk}^t_{\mathcal{U}}\}_{\mathcal{U}\in\mathsf{Usr}}$. For honest users $\mathsf{HUsr} \subseteq \mathsf{Usr}$, a block candidate bk^* of slot t is k-disclosed, if $\ell \geq t+k$ and for any honest users $\mathcal{HU} \in \mathsf{HUsr}$, $\mathsf{bk}^* \notin \mathsf{tree}^{t+k-1}_{\mathcal{HU}}$ and $\mathsf{bk}^* \in \mathsf{tree}^{t+k}_{\mathcal{HU}}$.*

The k-disclosure definition captures the notion that a block is known to all honest users within k time slots. The following lemma indicates that when a block extends a branch, and the branch is adopted as the highest-scored branch by at least one honest user, the block will be disclosed to all honest users.

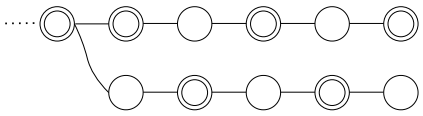
Lemma 1. *Assuming the network is δ-synchronous, a block candidate $\mathsf{bk}^* \in \mathsf{branch}^*$ is at most $(\delta+1)$-disclosed, if there exists an honest user $\mathcal{HU} \in \mathsf{HUsr}$ with $\mathsf{tree}_{\mathcal{HU}} = (V_{\mathcal{HU}}, E_{\mathcal{HU}})$ that satisfies $s_{acc}(\mathsf{branch}^*) = \max_{\mathsf{branch}^t}\{s_{acc}(\mathsf{branch}^t)\}$ where $\mathsf{branch}^t \subseteq \mathsf{tree}_{\mathcal{HU}}$ denotes the branch set in slot t.*

Proof. If bk^* is created by an honest user, it is 0-disclosed by the definition of 0-synchronous. If bk^* is created by the adversary, we assume an honest user $\mathcal{HU}$ adopts branch^* as its highest-scored branch of slot t and $\mathsf{bk}^* \in \mathsf{branch}^*$. Hence, in slot $t+1$, $\mathcal{HU}$ creates a block candidate $\mathsf{bk}^{t+1}_{\mathcal{HU}}$. Since $\mathsf{bk}^{t+1}_{\mathcal{HU}}$ is 0-disclosed, all honest users receive $\mathcal{HU}$'s extended path $\mathsf{branch}^*||\mathsf{bk}^{t+1}_{\mathcal{HU}}$ in slot $t+1$. Therefore, bk^* is added to their user-tree. The proof for an arbitrary δ follows similarly.

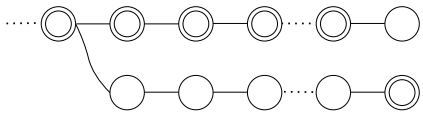
Directly from Lemma 1, we have the following proposition for disclosing the highest-scored branch in the master-tree to all honest users.

Proposition 1. *Assuming the network is δ-round synchronous, given a master-tree of slot ℓ, we denote the highest-scored branch of slot $\ell-(\delta+1)$ with branch^*. For any honest user $\mathcal{HU}$ and its user-tree $\mathsf{tree}^{\ell}_{\mathcal{HU}}$, $\mathsf{branch}^* \subseteq \mathsf{tree}^{\ell}_{\mathcal{HU}}$.*

Chain stability. Proposition 1 only indicates that the highest-scored branch of each time slot will eventually be *known* to all honest users. However, in two conjunctive slots, the selected highest-scored branches may be different from each other, *e.g.*, a good-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 being unstable and can harm the persistence in two ways (In the illustration, circle denotes the blocks on the branches, and double circle denotes that 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 2a); (2) if the selected branches of different slots have too many distinct blocks, the block history can get reset (Figure 2b). In both cases, invalid bids and transactions in the unstabilized blocks may disturb the trading 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. 2: Intuition of Chain Instability.

In order to tackle this problem, we first define divergence between branches and branch viability. Instead of length as in [14], our definitions depend on branch's score.

Definition 11 (Divergence and viability). *Let* mtree^ℓ *be a master-tree of time slot* ℓ*. In any slot* $t \leq \ell$*, for any two branches* $\mathsf{branch}_1, \mathsf{branch}_2$*, the divergence of* branch_1 *and* branch_2 *is given by* $\mathrm{div}(\mathsf{branch}_1, \mathsf{branch}_2) = |s_{acc}(\mathsf{branch}_1) - s_{acc}(\mathsf{branch}_2)|$*. Moreover, let* chain^t *be the highest-scored branch of slot* t*, a branch* $\mathsf{branch}^t \neq \mathsf{chain}^t$ *is viable, if* $\mathrm{length}(\mathsf{branch}^t) = \mathrm{length}(\mathsf{chain}^t) = \mathrm{height}(\mathsf{mtree}^t)$ *and* $\mathrm{div}(\mathsf{branch}^t, \mathsf{chain}^t) \leq \mathsf{smax} - \mathsf{smin}$ *where* $[\mathsf{smin}, \mathsf{smax}]$ *is the range of the scoring function.*

A branch in slot t is viable if it has the same length as the master-tree of slot t, and the divergence between the branch and the highest-scored branch is less than the length of a single block score's range. Hence, the viable branch could be extended with a single block with higher score and substitute the current selected chain. We formally define this situation as chain substitution in the following.

Definition 12 (Chain substitution). *Let* mtree^ℓ *be a master-tree of time slot* ℓ*. In a slot* $t \leq \ell$*, let* chain^t *be the highest-scored branch, and let a branch* $\mathsf{branch}^t \neq \mathsf{chain}^t$ *satisfies* $\mathrm{length}(\mathsf{branch}^t) = \mathrm{length}(\mathsf{chain}^t)$*. For two* $(t+1)$*-slot block candidates* $\mathsf{bk}_\mathsf{c}^{t+1}, \mathsf{bk}_\mathsf{b}^{t+1}$ *extending* chain^t *and* branch^t*, respectively. A chain substitution occurs when* $|s_{acc}(\mathsf{branch}_1) - s_{acc}(\mathsf{branch}_2)| \leq s(\mathsf{bk}_\mathsf{b}^{t+1}) - s(\mathsf{bk}_\mathsf{c}^{t+1})$*. Hence,* $s_{acc}(\mathsf{chain}^t || \mathsf{bk}_\mathsf{c}^{t+1}) \leq s_{acc}(\mathsf{branch}^t || \mathsf{bk}_\mathsf{b}^{t+1})$*.*

Notice that chain^t and $\mathsf{branch}^t || \mathsf{bk}_\mathsf{b}^{t+1}$ are the highest-scored branches in slot t and $t+1$, respectively. By Proposition 1, when assuming δ-synchronous, we need to set $t \leq \ell - (\delta + 2)$ so that these branches are disclosed. Moreover, a chain substitution happens only when the adversary finds a higher-scored block $\mathsf{bk}_\mathsf{b}^{t+1}$ to extend branch^t because honest users will work on chain^t, which indicates that the number of existing branches does not affect honest users' behavior. Thus, we infer there is only one viable branch without loss of generality. We denote the common part of branch_i and branch_j with $\mathsf{branch}_i \cap \mathsf{branch}_j$, which is a path from the genesis block bk_G to the last common block between the branches. Hence, we use $\setminus$ operation to denote the path on a branch from the first separate block, *i.e.*, $\mathsf{branch}_i \setminus \mathsf{branch}_j = \mathsf{branch}_i \setminus (\mathsf{branch}_i \cap \mathsf{branch}_j)$. Now, we analyze the probability of chain substitution with a parameter $\tau \geq 1$ such that $|\mathsf{chain}^t \setminus \mathsf{branch}^t| = \tau$. The following lemma provides a loose upper bound for the probability when assuming arbitrary score distribution and assuming the parameter function of the score accumulator to be $a(i) = c^i$ where c is a large enough constant value.

Lemma 2. *Assuming execution model in Section 2 and honest majority, let* mtree^ℓ *be a master-tree of time slot* ℓ*. Set the score accumulator's parameter function as* $a(i) = c^i$ *for a constant real number* c *and block's height in* mtree*. The probability of a chain substitution happening (in slot* $t+1 \leq \ell$*) for the highest-scored branch* chain^t *and a viable* branch^t *of slot* t *with* $|\mathsf{chain}^t \setminus \mathsf{branch}^t| = \tau \geq 1$ *is less than* $\mathsf{O}(c^{-\tau})$ *for a constant* c *(concrete bound with respect to arbitrary score distribution can be found in the following proofs).*

Proof. Recall Lemma 2: Assuming execution model in Section 2 and honest majority, let mtree^{ℓ} be a master-tree of time slot ℓ. Set the score accumulator's parameter function as $a(i) = c^i$ for a constant real number c and block's height in mtree. The probability of a chain substitution happening (in slot $t+1 \leq \ell$) for the highest-scored branch chain^t and a compatible branch^t of slot t with $|\mathsf{chain}^t \setminus \mathsf{branch}^t| = \tau \geq 1$ is less than $\mathsf{O}(c^{-\tau})$ for a constant c.

Estimating for $\tau = 1$. We first consider the situation where $\tau = 1$. Concretely, in slot t, we consider the highest-scored branch chain^t and a compatible branch branch^t, which get extended in slot $t+1$ by $\mathsf{bk}_{\mathsf{c}}^{t+1}$ and $\mathsf{bk}_{\mathsf{b}}^{t+1}$, respectively. Moreover, since $|\mathsf{chain}^t \setminus \mathsf{branch}^t| = 1$, chain^t and branch^t share a common path until slot $t-1$. Hence, we denote the last common block with $\mathsf{bk}^{t-1} \in \mathsf{chain}^t \cap \mathsf{branch}^t$. Moreover, the first distinct blocks on chain^t and branch^t are denoted with $\mathsf{bk}_{\mathsf{c}}^t \in \mathsf{chain}^t$ and $\mathsf{bk}_{\mathsf{b}}^t \in \mathsf{branch}^t$, respectively. When setting the parameter function of score accumulator in Definition 9 as $a(i) = c^i$, by chain substitution in Definition 12, for any $c > 0$, we have:

$$\begin{aligned} s(\mathsf{bk}_{\mathsf{c}}^t) &\geq s(\mathsf{bk}_{\mathsf{b}}^t), \\ c \cdot s(\mathsf{bk}_{\mathsf{c}}^t) + s(\mathsf{bk}_{\mathsf{c}}^{t+1}) &\leq c \cdot s(\mathsf{bk}_{\mathsf{b}}^t) + s(\mathsf{bk}_{\mathsf{b}}^{t+1}). \end{aligned} \tag{2}$$

Therefore, the probability of a chain substitution occurring for $\tau = 1$, denoted with Pr[Chain substitution for $\tau = 1$], equals to joint probability of events in Equation 2. That is, we need to estimate the following equation.

$$\Pr[s(\mathsf{bk}_{\mathsf{c}}^t) - s(\mathsf{bk}_{\mathsf{b}}^t) \geq 0 \wedge c \cdot (s(\mathsf{bk}_{\mathsf{c}}^t) - s(\mathsf{bk}_{\mathsf{b}}^t)) + s(\mathsf{bk}_{\mathsf{c}}^{t+1}) - s(\mathsf{bk}_{\mathsf{b}}^{t+1}) \leq 0] \tag{3}$$

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

$$\begin{aligned} Y^t &= X_{\mathsf{c}}^t - X_{\mathsf{b}}^t = s(\mathsf{bk}_{\mathsf{c}}^t) - s(\mathsf{bk}_{\mathsf{b}}^t), \\ Y^{t+1} &= X_{\mathsf{c}}^{t+1} - X_{\mathsf{b}}^{t+1} = s(\mathsf{bk}_{\mathsf{c}}^{t+1}) - s(\mathsf{bk}_{\mathsf{b}}^{t+1}). \end{aligned}$$

By the scoring oracle model, $(X_{\mathsf{c}}^t, X_{\mathsf{b}}^t, X_{\mathsf{c}}^{t+1}, X_{\mathsf{b}}^{t+1})$ are independent and follow the same distribution $\mathcal{D}_X = \mathcal{D}$ on $[\mathsf{smin}, \mathsf{smax}]$. Hence, Y^t and Y^{t+1} are independent, and their distribution are identical and *symmetric* [9] on $[\mathsf{smin} - \mathsf{smax}, \mathsf{smax} - \mathsf{smin}]$. We denote the distribution of Y^t and Y^{t+1} with $\mathcal{D}_Y$. Let the probability density function and the distribution function of $\mathcal{D}_Y$ be $f_Y(\cdot)$ and $F_Y(\cdot)$, respectively. For convenience of reading, we rewrite Equation 3 as:

$$\Pr[(Y^t \geq 0) \wedge (c \cdot Y^t + Y^{t+1} \leq 0)]. \tag{4}$$

In order to compute Equation 4, we consider the event: $\{Y^t = y \wedge Y^{t+1} \leq -cy\}$ for all $y \in [0, \mathsf{smax} - \mathsf{smin}]$. Let $r = \mathsf{smax} - \mathsf{smin}$. By Y^t is independent of

Y^{t+1}, we have:

$$\begin{aligned}\text{Equation 4} &= \int_0^r \Pr[(Y^t = y) \wedge (Y^{t+1} \leq -cy)]dy \\ &= \int_0^r \Pr[Y^t = y] \cdot \Pr[Y^{t+1} \leq -cy]dy \\ &= \int_0^r f_Y(y)F_Y(-cy)dy \\ &= \int_0^r \int_{-r}^{-cy} f_Y(y)f_Y(t)dtdy. \end{aligned} \tag{5}$$

Here, we set a upper-bound for Equation 5 by setting $f_Y(\cdot)$ with two coefficients $c_1, c_2 > 0$ as follows.

$$f_Y(y) \begin{cases} \leq c_1 \cdot e^{-c_2 \cdot y^2}, & \text{if } y \in [-r, r], \\ = 0, & \text{otherwise.} \end{cases} \tag{6}$$

Here, we can estimate the range of c_1 and c_2 because $f_Y(y)$ is a probability density function, *i.e.*, it satisfies $\int_{-r}^{r} f_Y(y)dy = 1$. Hence, $\int_{-\infty}^{\infty} c_1 \cdot e^{-c_2 \cdot y^2} dy \geq 1$, which is $\frac{c_1^2}{c_2} \geq \pi^{-1}$. By scaling Equation 5 with $f_Y(\cdot)$, we have:

$$\text{Equation 5} \leq \int_0^{\infty} \int_{-\infty}^{-cy} e^{-y^2} \cdot e^{-t^2} dtdy = \frac{{c_1}^2}{2c_2} \cdot \tan^{-1}(\frac{1}{c}). \tag{7}$$

Therefore,

$$\Pr[\text{Chain substitution for } \tau = 1] \leq \frac{{c_1}^2}{2c_2 \cdot c} \cdot \tan^{-1}(1/c) \leq \min\{1, \frac{{c_1}^2}{2c_2 \cdot c}\},$$

for any $c > 0$. In order to obtain meaningful result, we set $c > {c_1}^2/2c_2$, *i.e.*, $\Pr[\text{Chain substitution for } \tau = 1] \leq {c_1}^2/2c_2 \cdot c$.

Notice that the analysis of $\tau = 1$ works for any 1-slot chain substitution, *i.e.*, if we only consider the slots when a substitution happens. Hence, the probability of conjunctive chain substitutions for $\tau \geq 1$ slots is no greater than $\left(\frac{c_1^2}{2c_2}\right)^{\tau} \cdot c^{-\tau}$.

For $\tau > 1$. For branches chain^t and branch^t where $|\mathsf{chain}^t \setminus \mathsf{branch}^t| = \tau > 1$, we consider the situation where the branches share the same $(t-\tau)$-slot block $\mathsf{bk}^{t-\tau}$. Because a block can only extend at most one branch, $\mathsf{bk}^{t-\tau}$ is the last shared block between chain^t and branch^t. Hence, we denote the two sequences of blocks on chain^t and branch^t during slots $\{t-\tau+1, \ldots, t\}$ with $\mathsf{BKs_c} = \{\mathsf{bk_c}^{t-\tau+1}, \ldots, \mathsf{bk_c}^t\}$ and $\mathsf{BKs_b} = \{\mathsf{bk_b}^{t-\tau+1}, \ldots, \mathsf{bk_b}^t\}$ where $\mathsf{BKs_c} \cap \mathsf{BKs_b} = \phi$.

Moreover, we only consider the instability between chain^t and branch^t, *i.e.*, let chain^i be the highest-scored branch in slot $i \in \{t-\tau+1, \ldots, t\}$, $\mathsf{chain}^i \subseteq \mathsf{chain}^t$ or $\mathsf{chain}^i \subseteq \mathsf{branch}^t$. Therefore, we assume, without loss of generality, blocks in

$\mathsf{BKs_c}$ are selected in τ_1 slots, and blocks in $\mathsf{BKs_b}$ are selected in τ_2 slots where $\tau = \tau_1 + \tau_2$. Since in slot t, the selected block is bk_c^t, we now consider an extreme situation where all the blocks during the unstable period are selected from $\mathsf{BKs_c}$, *i.e.*, $\tau_1 = \tau$. The situation when $\tau_1 < \tau$ follows similarly. In the following, we evaluate the probability.

By the same methodology of $\tau = 1$, we rewrite Equation 4 for $\tau > 1$ as:

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

where $Y^i = X_\mathsf{c}^i - X_\mathsf{b}^i$ for any $i \in \{t-\tau+1, \ldots, t+1\}$ are independent and distributed identically with $\mathcal{D}_Y$ on $[\mathsf{smin}-\mathsf{smax}, \mathsf{smax}-\mathsf{smin}]$. Hence, Equation 8 equals to:

$$\begin{aligned}
&\int_0^r \int_{r_{t-\tau+i}}^r \cdots \int_{r_t}^r \Pr\left[\left(\bigwedge_{i\in\{t-\tau+1,\ldots,t\}} Y^i = y_i\right) \wedge \left(Y^{t+1} \leq -\sum_{i=1}^{\tau} c^i \cdot y_{t+1-i}\right)\right] dy_{t-\tau+1} \cdots dy_t \\
&= \int_0^r \cdots \int_{r_t}^r \Pi_{i=t-\tau+1}^t \Pr[Y^i = y_i] \cdot \Pr\left[Y^{t+1} \leq -\sum_{i=1}^{\tau} c^i \cdot y_{t+1-i}\right] dy_{t-\tau+1} \cdots dy_t \\
&= \int_0^r \cdots \int_{r_t}^r \Pi_{i=t-\tau+1}^t f_Y(y_i) \cdot F_Y(-\sum_{i=1}^{\tau} c^i \cdot y_{t+1-i}) dy_{t-\tau+1} \cdots dy_t \\
&= \int_0^r \cdots \int_{r_t}^r \int_{-r}^{-\sum_{i=1}^{\tau} c^i \cdot y_{t+1-i}} \Pi_{i=t-\tau+1}^t f_Y(y_i) \cdot f_Y(t) dt dy_{t-\tau+1} \cdots dy_t. \quad (9)
\end{aligned}$$

We denote the lower bound of random variable $Y_{t-\tau+i}$ for any $i \in \{, \ldots \tau\}$ as $r_{t-\tau+i}$. Hence $r_{t-\tau+i} = -\sum_{j=1}^{i} c^j \cdot y_{t-\tau+j}$.

Next, We use a small trick to scale Equation 9. Since Y^t's lower bound r_t derives from $\sum_{i=0}^{\tau-1} c^i \cdot y_{t-i} \geq 0$, and Y^{t+1}'s upper bound is $-\sum_{i=1}^{\tau} c^i \cdot y_{t+1-i}$, we can scale with $y_i = -r$ for any $i \in \{t-\tau+\mathbf{2}, \ldots, t\}$ so that $y_{t-\tau+1} \geq \frac{c^{\tau-1}-1}{c^{\tau-1}(c-1)}$ and $Y^{t+1} \leq -c^\tau \cdot (y_{t-r+1} - \frac{c^{\tau-1}-1}{c^{\tau-1}(c-1)})$. Moreover, by setting $y' = y_{t-r+1} - \frac{c^{\tau-1}-1}{c^{\tau-1}(c-1)}$ and by $\int_{-r}^{r} f_Y(y_i) dy_i \leq 1$ for any $i \in \{t-\tau+1, \ldots, t\}$, we have:

$$\begin{aligned}
\text{Equation 9} &\leq \int_0^r \int_{-r}^{-c^\tau \cdot (y_{t-r+1} - \frac{c^{\tau-1}-1}{c^{\tau-1}(c-1)})} f_Y(y_{t-\tau+1}) f_Y(t) dt dy_{t-\tau+1} \\
&\leq \int_{-r}^r \int_{-r}^{-c^\tau \cdot y'} f_Y(y') f_Y(t) dt dy'. \quad (10)
\end{aligned}$$

Finally, by the bound of $f_Y(\cdot)$ given in Equation 6, we have:

$$\text{Equation 10} \leq \int_{-\infty}^{\infty} \int_{-\infty}^{-c^\tau y'} e^{-y'^2} \cdot e^{-t^2} dt dy' = \frac{{c_1}^2}{c_2} \cdot \tan^{-1}(\frac{1}{c^\tau}) \leq \frac{c_1^2}{c_2} \cdot c^{-\tau}. \quad (11)$$

Therefore, combining the discussion above, we conclude that the probability of a chain substitution happening for chain^t and branch^t in slot $t+1$ where $|\mathsf{chain}^t \setminus \mathsf{branch}^t| = \tau \geq 1$ is less than $\max\left\{\left(\frac{c_1^2}{2c_2}\right)^{\tau} \cdot c^{-\tau}, \frac{c_1^2}{c_2} \cdot c^{-\tau}\right\}$. Here, $c_1, c_2 > 0$ is the coefficient in the subtraction distribution of the scoring function/oracle's output distribution. We can set the scoring function/oracle's output to arbitrary distribution as long as $\frac{c_1^2}{c_2} \geq \pi^{-1}$ and $c \geq \max\{1, \frac{c_1^2}{c_2}\}$.

Hence, we have the following theorem on persistence.

Theorem 1 (Persistence). *Given the settings from in Lemma 1 and Lemma 2, the probability of our protocol achieving persistence on the* $\mathsf{chain}^{\ell,\rho}$ *in time slot* ℓ *is no less than* $\Omega(1 - c^{-\rho+\delta+1})$ *where* $\rho \geq \delta + 1$ *is a parameter and* $\mathsf{chain}^{\ell,\rho}$ *denotes the branch from* bk_G *to the block on* chain^{ℓ} *of slot* $\ell - \rho$.

Proof. Suppose persistence is violated for parameter $\rho \geq \delta+1$. It follows that, in two different slots $\ell_1 \leq \ell_2$, the selected chain of ℓ_1 up-to slot $\ell_1 - \rho$, $\mathsf{chain}_1^{\ell_1,\rho}$, is *NOT* the prefix of the selected chain of ℓ_2, chain_2. Hence, $\mathsf{chain}_1^{\ell_1,\rho} \setminus \mathsf{chain}_2 \neq \phi$, and there exists a block $\mathsf{bk}_1^{\ell_1-\rho'}$ such that $\mathsf{bk}_1^{\ell_1-\rho'} \notin \mathsf{chain}_2$ where $\rho' \geq \rho$.

By Proposition 1, the sub-branches on the chains $\mathsf{chain}_1^{\ell_1,\rho}$ and $\mathsf{chain}_2^{\ell_1,(\delta+1)}$ are disclosed to all honest users. Notice that we consider slot $\ell_1 - (\delta + 1)$ for chain_2 because $\ell_1 \leq \ell_2$ and thus, $\ell_1 - (\delta+1) \leq \ell_2 - (\delta+1)$. Because both chains are selected as the highest-scored chain in some slots and $\mathsf{chain}_1^{\ell_1,\rho} \setminus \mathsf{chain}_2^{\ell_2} \neq \phi$, a chain substitution must happen in a slot after $\ell_1 - (\delta + 1)$. Otherwise, chain_1 cannot be selected in slot ℓ_1.

Therefore, chain_1 and chain_2 separate from each other in slot $\ell_1 - \rho'$ with $\rho' \geq \rho$ and chain_2 substitutes chain_1 after $\ell_1^(\delta+1)$. By Lemma 2, the probability of such a chain substitution happens with probability less than $\mathsf{O}(c^{-\rho+\delta+1})$. Thus, the probability of $\mathsf{chain}_1^{\ell_1,\rho}$ *BEING* the prefix of chain_2 is no less than $\Omega(1 - c^{-\rho+\Delta+1})$ where c is a constant real value given in the proof of Lemma 2.

6.2 Achieving Block-Liveness

For completeness, we show that our protocol satisfies block-liveness following directly from the execution model in the following theorem.

Theorem 2 (Block-liveness). *Assuming at least one honest participant in the protocol, we achieve block-liveness unconditionally under the time slot-based execution with a globally synchronized clock.*

The proof is fairly straightforward since the execution model is restricted. Briefly, assume only one participant is honest, then it generates block candidates with the locally kept bids. Given that it is unaware of any other block, the honest participant trivially extends the local chain with the block candidate.

7 Final Remarks

The starting point of our protocol is to implement P2P/ETS, therefore we generalized it to a *auction market*. Despite our constrained setting, *i.e.*, we assumed a δ-synchronous network, static corruptions of at most half participants equipped with synchronized clocks to generate reliable timestamps. Typically, this setting suits our needs for P2PETS. We advocate the energy trading infrastructure is already deployed and show a similar setting, *i.e.*, private energy grids in small communities. In such environments, our protocol can be a suitable option.

One component of our construction is the scoring function s. Its purpose is to apply a "notion of optimal" to the system. Among all the blocks available on each time slot, it chooses the "most optimal" choice to extend a 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 is of independent interest and out of the scope of this work.

In our analysis, the scoring function is replaced by a *scoring oracle* which assigns random scores for candidate blocks on 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 oracle, it is practical, since it can be easily used in practice with a random function. However, the adaptability of the adversary, early mentioned, is not possible since the adversary is not given a representation of the function.

Considering the scoring oracle in the early constrained but meaningful setting, we showed that our protocol has consistency and block-liveness when half of the participants are 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.

References

1. Raphaelle Akhras, Wassim El-Hajj, Michel Majdalani, Hazem M. Hajj, Rabih A. Jabr, and Khaled B. Shaban. Securing smart grid communication using ethereum smart contracts. In *16th International Wireless Communications and Mobile Computing Conference, IWCMC 2020, Limassol, Cyprus, June 15-19, 2020*, pages 1672–1678. IEEE, 2020.
2. Anthony B Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 2(3):244–263, 1970.
3. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.
4. Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*, page 23–41, Berlin, Heidelberg, 2019. Springer-Verlag.
5. Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*, page 23–41, Berlin, Heidelberg, 2019. Springer-Verlag.
6. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *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*, volume 10821 of *Lecture Notes in Computer Science*, pages 66–98. Springer, 2018.
7. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *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*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.
8. Juan A. Garay, Aggelos Kiayias, Nikos Leonardos, and Giorgos Panagiotakos. Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings*, volume 10770 of *Lecture Notes in Computer Science*, pages 465–495. Springer, 2018.
9. George C. Tiao George E.P. Box. *Bayesian Assessment of Assumptions 1. Effect of Non-Normality on Inferences about a Population Mean with Generalizations*, chapter 3, pages 149–202. John Wiley and Sons, Ltd, 1992.
10. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
11. Tomasz Górski and Jakub Bednarski. Modeling of smart contracts in blockchain solution for renewable energy grid. In Roberto Moreno-Díaz, Franz Pichler, and

Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2019 - 17th International Conference, Las Palmas de Gran Canaria, Spain, February 17-22, 2019, Revised Selected Papers, Part I*, volume 12013 of *Lecture Notes in Computer Science*, pages 507–514. Springer, 2019.

12. Mariusz Kaleta. Price of fairness on networked auctions. *J. Appl. Math.*, 2014:860747:1–860747:7, 2014.
13. Simon Holmgaard Kamp, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, Søren Eller Thomsen, and Daniel Tschudi. Weight-based nakamoto-style blockchains. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*, volume 12912 of *Lecture Notes in Computer Science*, pages 299–319. Springer, 2021.
14. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017.
15. Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 254–269. ACM, 2016.
16. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. `http://bitcoin.org/bitcoin.pdf`.
17. Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *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*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.
18. Harry Robertson. Ethereum transaction fees are running sky-high., 2021. "https://markets.businessinsider.com/news/currencies/ethereum-transaction-gas-fees-high-solana-avalanche-cardano-crypto-blockchain-2021-12".
19. G. Terry Ross and Richard M. Soland. A branch and bound algorithm for the generalized assignment problem. *Math. Program.*, 8(1):91–103, 1975.
20. George Roussas. *An introduction to probability and statistical inference*, pages 207–243. Elsevier, Academic Press, 12 2015.
21. Gavin Wood. Ethereum yellow paper, 2014.