# Minting Mechanisms for Blockchain – or – Moving from Cryptoassets to Cryptocurrencies*

Dominic Deuber[1], Nico Döttling[2], Bernardo Magri[1], Giulio Malavolta[1], and Sri Aravinda Krishnan Thyagarajan[1]

[1] *Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany*
[2] *CISPA Helmholtz Center, Germany*

November 15, 2018

**Abstract**

Permissionless blockchain systems, such as Bitcoin, rely on users using their computational power to solve a puzzle in order to achieve a consensus. To incentivise users in maintaining the system, newly minted coins are assigned to the user who solves this puzzle. A hardware race that has hence ensued among the users, has had a detrimental impact on the environment, with enormous energy consumption and increased global carbon footprint. On the other hand, proof of stake systems incentivise coin hoarding as players maximise their utility by holding their stakes. As a result, existing cryptocurrencies do not mimic the day-to-day usability of a fiat currency, but are rather regarded as crypto-assets or investment vectors.

In this work we initiate the study of minting mechanisms in cryptocurrencies as a primitive on its own right, and as a solution to prevent coin hoarding we propose a novel minting mechanism based on waiting-time first-price auctions. Our main technical tool is a protocol to run an auction over any blockchain. Moreover, our protocol is the first to securely implement an auction *without* requiring a semi-trusted party, i.e., where every miner in the network is a potential bidder. Our approach is generically applicable and we show that it is incentive-compatible with the underlying blockchain, i.e., the best strategy for a player is to behave honestly. Our proof-of-concept implementation shows that our system is efficient and scales to tens of thousands of bidders.

*This paper is part of the work of the Nuremberg Campus of Technology, a research cooperation of Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Technischen Hochschule Nürnberg Georg Simon Ohm, supported by the state of Bavaria.

## Contents

## 1 Introduction

Since its onset in 2008, Bitcoin [Nak08] has paved the way for the emergence and widespread acceptance of several hundred cryptocurrencies all over the globe, promising an era of borderless decentralised digital currency system. At the time of writing, there are over 1.7K cryptocurrencies with a total value of more than 260 Billion dollars. These currencies can be characterised according to their consensus mechanism, namely, Proof of Work (PoW), Proofs of Space, and Proof of Stake (PoS) among other variants.

Proof of Work based consensus systems, such as Bitcoin, rely on users solving a hard computational puzzle to achieve decentralised consensus on the state of the system. Since no efficient algorithm is known for solving such a puzzle, users have to rely on their computational power for an exhaustive search of the solution. This process is often referred to as "mining". Miners work to maintain the system by validating transactions and a reward is assigned to the miner who solves the puzzle first. Apart from the reward, the miner also collects fees from the transactions he validated. This incentive mechanism has led to a hardware race [Tay13, Pec13, Min17], which has resulted in enormous energy demands and environmental problems [OM14]. As an undesirable consequence, mining has now become feasible only to major investors who can afford powerful mining hardware [Poo17].

To mitigate the problems mentioned above, the community investigated alternative consensus mechanisms, based on more energy-efficient resources. Proof of Space proposals [ABFG14, DFKP15] are based on disk space as the consensus resource, while Proof of Useful Work [Kin13, MJS$^+$14] achieve consensus by performing work that may turn useful in practice. Proof of Stake (PoS) [KN12, KRDO17] systems rely on the rationality of a stakeholder in the system to behave honestly due to the risk of devaluing the currency. In PoS, the consensus leader is chosen solely based on her stake in the system. This mechanism has the appealing feature of not being "wasteful" as no resource is consumed to achieve a consensus.

**An Unfulfilled Promise.** The meteoric rise in the value of virtually all cryptocurrencies has caused large scale *hoarding* of coins: Users are reluctant to trade their coins, fearing a sudden surge (due to high volatility) in value of the currency. Bitcoin is speculated to be the new *digital gold*, rather than a currency for day-to-day usage [Met]. Bitcoin's fixed cap on the supply of new coins and energy demanding PoW mining, has resulted in people viewing it as an investment vector [ass, Kot] instead of a currency. On the other hand, in PoS based systems, the stakeholders are incentivised to hold their stake to maximise their probability of becoming the consensus leader and collect new coins and/or transaction fees. In both the above scenarios, the system is affected by significant deflation of coins, thereby discouraging users to engage in day-to-day trading (in contrast to fiat currencies).

**Towards Cryptocurrencies.** The relation between money supply and hoarding is a well studied topic in economic theory. Tsiang [Tsi89] advocates for a moderate inflation as a countermeasure for the stagnation of money. Several other works in the literature [Sat11, Hum07, Woo11, OS90] extensively study a steady inflation (in the form of increase in money supply) as a deterrent for money hoarding and as an incentive for trading. Predictably, fixing the supply of coins (as in Bitcoin) and incentivising stake-holding (as in PoS), have the opposite effect. This issue may hinder the long-term viability of cryptocurrencies as an alternative to fiat currency systems.

**State of the Art Minting Mechanisms.** While consensus seems to be a better understood problem [KRDO17, GKL15, PSS17, GKL17] given the current state of affairs, there is no unified solution for the introduction of new coins in a cryptocurrency. Current folklore approaches are either energy expensive (such as PoW-based systems) or incentivise hoarding of stakes (such as PoS-based approaches). Surprisingly, this problem has hardly received any attention and, to the best of our knowledge, there is no rigorous treatment of minting mechanisms in cryptocurrencies.

The problem of a secure minting mechanism shares several challenges with those that exist for achieving consensus. Above all, the main difficulty lies in the prevention of *Sybil* attacks: A malicious user may spawn arbitrarily many identities and exploit the benefits of the newly created users simultaneously. Trivial solutions, such as randomly assigning newly minted coins to a user in the system are vulnerable to this family of attacks. In a fully distributed scenario, Sybil attacks are hard to prevent and may lead to catastrophic consequences [Goo].

Most of the current systems (such as PoW [Nak08, GKL15] or proofs of space [DFKP15, ABFG14]) have integrated the distribution of new coins with the consensus mechanism: The miner who proposes the new block, is also rewarded with the newly minted coins. However, the brute-force approach (of PoW especially) to obtain the reward has resulted in a hardware race among the miners [Min17] and subsequent increase in the difficulty of mining. PoS-based system either do not mint new coins at all (fixed cap) [KRDO17], or assign the new coins to the consensus leader [KN12]. As discussed above, this invariably incentivises coin hoarding by the stakeholders and promotes the deflation of the currency. To the best of our knowledge, the problem of a unified minting mechanism for cryptocurrencies is still open.

**Decoupling Minting from Consensus.** In this work we initiate the study of minting mechanisms as a primitive on its own right and we propose a new protocol based on waiting-time auctions. Any user in the system only requires a small amount of coins to compete for the newly minted coins. As a result, the system mitigates coin hoarding and incentivises the participation of regular users, as they can compete with large investors. In a nutshell, our system rewards the user who is willing to "wait the longest", after the user has waited for that amount of time. Under the assumption that users cannot stack the time at their disposal, pooling resources does not increase the chances of receiving the new coins (thereby preventing sybil attacks).

On a conceptual level, we suggest a hybrid approach for cryptocurrencies, where the minting mechanism is decoupled from the consensus. The consensus is only incentivised by the collection of transactions fees, while the minting of new coins in the system is carried out by the minting mechanism, with its own set of rules. This makes our proposal directly applicable to any form of consensus and, in particular, to PoS-based systems.

## 1.1 Our Contributions

Our contributions can be summarised as follows.

1. We initiate the rigorous treatment of minting mechanisms in cryptocurrencies and we analyse the pitfalls of folklore solutions. We introduce the concept of *utility-preserving* stake allocation (section 4), on the same spirits of Pareto efficiency. Informally, this property states that in a utility-preserving system, stakeholders can trade their stake without affecting their chances of obtaining newly minted coins. Using this property we analyse and show that coin hoarding is in fact incentivised in a PoS-based minting mechanism where new coins are assigned to the consensus leader. To backup this claim, we present empirical evidence that PoS-based systems have a significantly lower number of transactions between users.
2. We propose a new minting mechanism based on waiting-time auctions (section 3) and we show that it is incentive-compatible with the underlying blockchain (subsection 5.4), i.e., following honestly the protocol is the Nash equilibrium strategy for rational miners on the blockchain system. We also formally show that our mechanism is utility-preserving in its stake allocation, and therefore mitigates the problem of coin hoarding. Informally, this is

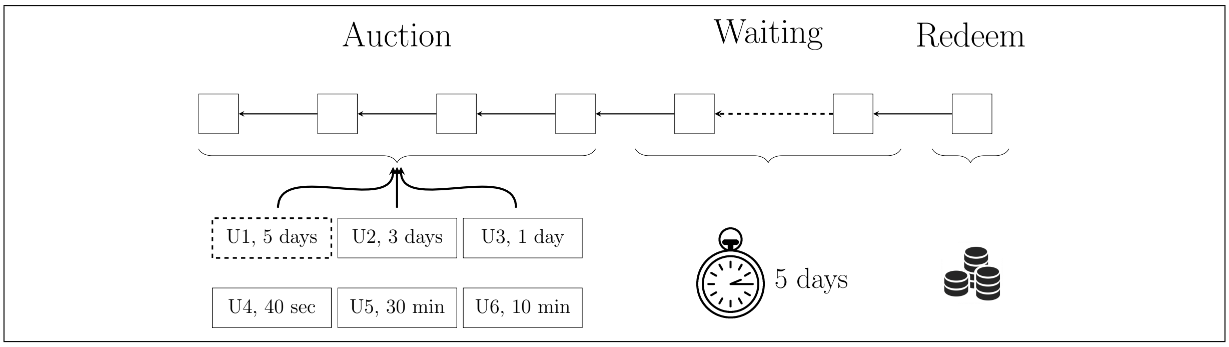

Figure 1: Waiting-time based rewarding where user U1 is prepared to wait the longest (5 days), and obtains the reward after waiting for 5 days.

because the stakeholder needs only a token to participate in a minting round, while the rest of the coins are free to be traded.

3. On a technical level, we present a cryptographic construction (subsection 5.3) for realising a first-price auction on top of a blockchain. Our protocol does not require any additional interaction other than what is required by the underlying blockchain, and does not rely on any *semi-trusted party*. Our solution is the first where every miner in the network is a potential bidder. This is in strong contrast with previous proposals that assume the existence of a semi-trusted auctioneer to collect bids and announce the winner.
4. We demonstrate the scalability of our approach with a proof-of-concept implementation (section 6) of our construction and a thorough performance analysis. The system can be scaled to support thousands of bidders per block with a reasonable block size (8MB) while leaving more than two-thirds of the block free for standard transactions.

## 1.2 Our Solution

To circumvent the problem of Sybil attacks, the minting mechanism must rely on some quantifiable resource. On that regard, we identify *time* to be such a resource. The time that we consider here is the physical time one has in her future, or in other words, the notion of "from now on". Our minting mechanism leverages the observation that the time at one's disposal is (roughly) equal across the set of users and *cannot be combined with the time of other users.*

**Minting Mechanism.** We describe our mechanism under the assumption of the existence of an underlying blockchain system. Specifically, our protocol can be built on top of any public transaction ledger whose consensus relies solely on transaction fees as incentive. Our protocol implements a sequential first-price auction, does not require an auctioneer, and the miners can actively participate in the protocol and compete for the rewards. We leverage rational arguments to show that the best strategy for every user is to simply follow the protocol specification. Figure 1 gives a pictorial overview of one full round of our minting mechanism, that consists of an auction round, waiting period and redeem period. Each auction round in itself consists of three phases:

1. At periodic intervals users engage in a first-price auction where the item being auctioned are $R$ newly minted coins. The bidding phase for the auction spans through $\alpha$ blocks where every user willing to participate posts a bid transaction with a concealed bid. The bid here is the amount of physical time units the user is willing to wait in order to obtain the minted coins. To be eligible to participate, a user is required to "lock" some fixed amount $Q$ of his coins (called *token of participation* or *participation token*) for the entire duration of the auction.

2. Once the bidding phase is over, the protocol allocates $\beta$ blocks for users to broadcast the unveil information of their bids. We call these $\beta$ blocks the opening phase.
3. After the opening phase, miners can open all the posted bids (using the corresponding unveil information) and determine the winner of the auction. A mint transaction is then generated assigning $R$ newly minted coins to the winner of the auction, that can be redeemed only after the time corresponding to her bid has elapsed. All users can unlock their token of participation $Q$ after the auction round is over, except the winner, who only gets back $Q$ together with the minted coins.

**Cryptographic Implementation.** As a first (flawed) attempt, consider a protocol in which every bidder posts a transaction with a commitment *com* to their bid, then later in an opening phase they post the unveil $r$, and the winner can be publicly determined. The challenge that arises here is how to deal with the case where a player does not post the opening to their bid. If there is a mechanism in place to actively prevent this behaviour, e.g. by excluding this player from the auction and determining the winner among the other bidders, then this constitutes an incentive for miners to suppress the openings of higher bidders. On the other hand, if no such mechanism is in place and the auction is aborted after a certain time, then a single bidder can prevent the minting of new coins.

Current proposals for running auctions over blockchains [KMS+16, BK] make clever usage of smart contracts but assume the existence of a semi-honest party that is entrusted to run the auction contract. This party is aware of the bidders' inputs and it is trusted to not disclose or manipulate that information. In our setting, every miner is a potential bidder, and at the same time might be in charge of including the bids of other users in the current block. This opens the door to "bid suppression" attacks where such a miner selectively discards some bids to increase his chances of winning.

To deal with these apparently conflicting requirements, we propose a cryptographic solution where each round of the auction can be completed even if players go offline after the bidding phase. Our protocol requires players to embed the unveil information $r$ in a time-lock puzzle *tlp* during the bidding phase. Time-lock puzzles ensures that their payload is hidden for a stipulated amount of time but can be opened once this amount of time has elapsed. This means that bids remain concealed until the end of the bidding phase but can be efficiently recovered in case *a player does not publish* the unveil of the corresponding commitment (i.e., the player goes offline). This effectively eliminates the need for a trusted party in the execution of the auction over the blockchain. We stress however, that in a rational run of the protocol the time-lock puzzles are *never* required to be solved as the bidders reveal the bids during the opening phase. Time-lock puzzles are only used as a *deterrent* against malicious bidders who refuse to open their bids.

**Towards a Scalable System.** One of the major challenges tied to our approach is to ensure that the system can be scaled to support a significant amount of bidders without affecting the transaction throughput dramatically. A subtle issue with what was discussed above is that there is no mechanism in place to ensure that the time-lock puzzle *tlp* contains a valid unveil $r$ of the associated commitment *com*. An obvious solution would be to include a non-interactive zero-knowledge proof [BFM88] of consistency. However, attaching a proof to each bid would have a critical impact on the size of transactions (and consequently of the blocks) and on the computational effort of the bidders.

We instead adopt a different strategy, where players are *not* required to prove the well-formedness of their bids. When miners encounter a puzzle that contains some information $r$ which is *not* a valid unveil for *com*, they can "steal" the participation token of that user by publishing a recovery proof of $r$. Recovery proofs [BN00, Pie18] allow one to publicly verify that a certain payload $r$ is contained in a time-lock puzzle *tlp* exponentially faster than simply solving the puzzle. It can then be publicly verified that $r$ is not a valid opening of *com*. This mechanism discourages players from posting malformed bids, since they will eventually be caught and lose their participation token. The advantage of this solution is that cryptographic proofs are invoked only for the few cases where the bids are malformed, as opposed to having a consistency proof for *each* bid transaction.

**Formal Analysis.** Our protocol can be formally modelled as a first-price sequential auction with sealed bids and we leverage state-of-the-art results on sequential auctions [LST12] to show that our rewarding mechanism has a Nash equilibrium on the amount of time units that a user should bid in each round of the auction. Then we analyse the utility-preserving stake allocation of our system and we show that our minting mechanism incentivises stake trading. In contrast, for folklore minting solutions, *all* stake allocations are *not* utility-preserving, which does not promote coin circulation and lead to stake hoarding. Finally, we prove that our mechanism is incentive-compatible with the underlying blockchain, i.e., honestly following the protocol is the Nash equilibrium strategy for rational miners on the blockchain system.

**Implementation.** As a proof-of-concept implementation of our system we build an entire blockchain system coupled with our minting mechanism. Considering a bidding phase of 10 blocks and blocks of size 8MB, we can fit more than 10K bids in a single auction round and still leave around 70% of the block's capacity free for standard transactions. To produce a proof for a mint transaction including 750 bids, the system takes less than 3 minutes, and the verification is almost instant, as we show in subsection 6.3. We remark that, at the time of writing, there are roughly 10K Bitcoin nodes around the globe, and our protocol can easily support these numbers without much overhead on the underlying blockchain.

## 2 Preliminaries

Throughout this work we denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\mathsf{in})$ the output of the algorithm $\mathcal{A}$ on input $\mathsf{in}$.

### 2.1 Cryptographic Building Blocks

Below we recall the cryptographic primitives used in our protocol and we refer the reader to Appendix A for formal definitions.

**Non-interactive CCA-Commitment Schemes.** A non-interactive tagged commitment scheme consists of a pair of randomised algorithms: a setup $\mathsf{Setup}(1^\lambda)$, that takes as input the security parameter and outputs a common reference string $\mathtt{crs}$, and a commitment $\mathsf{Commit}(\mathtt{crs}, \mathtt{addr}, m; r)$ that takes as input a common reference string $\mathtt{crs}$, a tag/identity $\mathtt{addr}$, a message $m$ and random coins $r$ and outputs a commitment *com*. Loosely speaking, the commitment *com* should hide the message $m$, and it should be infeasible for anyone to show a valid set of coins $r'$ that such that $\mathsf{Commit}(\mathtt{crs}, \mathtt{addr}, m'; r) = com$ for a different message $m'$. Additionally, for such schemes it is not possible to "maul" commitments for one tag into commitments for another tag. Such commitment schemes can be constructed from standard SHA-256 commitments in the random oracle model [BR93].

**Time-Lock Puzzles.** A time-lock puzzle allow one to conceal a value for a certain amount of time. The puzzle generation algorithm $\mathsf{PGen}(1^\lambda, \mathbf{T}, m)$ takes as input a security parameter, a hardness-parameter $\mathbf{T}$ and a message $m$, and outputs a puzzle *tlp*. The puzzle *tlp* can be cracked using the solving algorithm $\mathsf{PSolve}(tlp)$, which outputs the message $m$ and a recovery proof $\pi$. The proof can be verified with the corresponding verification algorithm $\mathsf{PVer}(tlp, m, \pi)$.

Time-lock puzzles guarantee that a puzzle can be solved in polynomial time, but strictly higher than $\mathbf{T}$. Additionally, verifying a recovery proof shall be exponentially faster then solving the puzzle. The first and to date only efficient candidate construction of time-lock puzzles was given by Rivest, Shamir and Wagner [RSW96] and is based on a variant of the RSA assumption. Boneh and Naor [BN00] showed how to compute a recovery proof such that its verification is exponentially faster than solving the puzzle. Subsequently, Pietrzak in his recent work [Pie18] proposed an improvement in the efficiency of the recovery proof.

**Succinct Non-interactive Arguments.** Let $R : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be an *NP*-witness-relation with corresponding *NP*-language $\mathcal{L} := \{x : \exists w \text{ s.t. } R(x,w) = 1\}$. A succinct non-interactive argument (SNARG) [Mic00] system for $R$ is initialised with a setup algorithm $\mathsf{crsGen}(1^\lambda)$ that, on input the security parameter, outputs a common reference string $\texttt{crs}$. A prover can show the validity of a statement $x$ with a witness $w$ by invoking $\mathsf{P}(\texttt{crs}, x, w)$, which outputs a proof $\pi$. The proof $\pi$ can be efficiently checked by the verification algorithm $\mathsf{V}(\texttt{crs}, x, \pi)$. We require a SNARG system to be sound, in the sense that it is hard for any prover to convince a verifier of a false statement, and proofs to be succinct (of size independent of $x$ and $w$).

## 2.2 Execution Model

In the following we define the notation for our protocol executions. Our definitions follow along the same lines of [PS17].

A protocol refers to an algorithm for a set of interactive Turing Machines (also called nodes) to interact with each other. The execution of a protocol $\Pi$ that is directed by an environment/outer game $\mathcal{Z}(1^\lambda)$, which activates a number of parties $\mathcal{P} = \{p_1, \ldots, p_n\}$ as either honest or corrupted parties. Honest parties would faithfully follow the protocol's prescription, whereas corrupt parties are controlled by an adversary $\mathcal{A}$ which reads all their inputs/message and sets their outputs/messages to be sent.

- A protocol's execution proceeds in rounds that model atomic time steps. At the beginning of every round, honest parties receive inputs from an environment $\mathcal{Z}$; at the end of every round, honest parties send outputs to the environment $\mathcal{Z}$.
- $\mathcal{A}$ is responsible for delivering all messages sent by parties (honest or corrupted) to all other parties. $\mathcal{A}$ cannot modify the content of messages broadcast by honest parties.
- At any point, $\mathcal{Z}$ can corrupt an honest party $j$ which means that $\mathcal{A}$ gets access to its local state and subsequently, $\mathcal{A}$ controls party $j$. (In particular, this means we consider a model with *erasures*: Random coin tosses that are no longer stored in the local state of $j$ and therefore are not visible to $\mathcal{A}$.)
- At any point of the execution, $\mathcal{Z}$ can uncorrupt a corrupted party $j$, which means that A no longer controls $j$. A party that becomes uncorrupt is treated in the same way as a newly spawning party, i.e., the party's internal state is re-initialised and then the party starts executing the honest protocol no longer controlled by $\mathcal{A}$.

Note that a protocol execution can be randomised, where the randomness comes from honest parties as well as from $\mathcal{A}$ and $\mathcal{Z}$. We denote by $\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \lambda)$ the randomly sampled execution trace. More formally, $\mathsf{view}$ denotes the joint view of all parties (i.e., all their inputs, random coins and messages received, including those from the random oracle) in the above execution; note that this joint view fully determines the execution. For convenience, we denote by $\mathsf{view}_\delta$ the state of the execution at round $\delta$ and we define the following two functions: On input an execution $\mathsf{view}_\delta$ at a certain round $\delta$, the function $\mathsf{public}$ returns the public state of the protocol, i.e., the information available to all machines, whereas honest returns the set of honest parties at round $\delta$.

### 2.3 Rational Security

Here we give a brief overview of the notion of rational players, following the definitions of [HP15]. Every player is characterised by some payoff (or utility) function $u$. In any protocol (game), utility represents the motivations of players. A utility function for a given player assigns a number for every possible outcome of the protocol with the property that a higher number implies that the outcome is more preferred. A *rational* player wishes to maximise her utility.

Every player is also equipped with a strategy function. A strategy function takes as input the view of the player so far and outputs its next action. Rational players will choose from the strategies available to them the one that results in the most preferred outcome. Note that the strategies and the protocol can have potential randomness which invokes a certain distribution over the outcomes of the protocol. We define the utility of a distribution as the the expected value of the utility of an outcome drawn from that distribution.

Let $Z$ be a family of subsets of the set of players for a game $G$. We say that a set of strategies $\mathbf{s}$ constitutes a $Z$-coalition-safe $\varepsilon$-Nash-equilibrium, if no coalition of players from a set $Z$ can gain more than $\varepsilon$ in payoff when deviating from $\mathbf{s}$ when playing $G$.

A mediated game is one in which a trusted party, the mediator, takes inputs from players, computes a function and provides outputs to the players. Following [HP15] we say that a protocol $\Pi$ implements a mediator $\mathcal{F}$ if it holds for any admissible environment/outer gamer $\mathcal{Z}$ that if it is an equilibrium strategy to truthfully provide inputs to $\mathcal{F}$ in game $\mathcal{Z}$, then it is an $\varepsilon$-equilibrium strategy to honestly execute protocol $\Pi$ in $\mathcal{Z}$, where $\varepsilon$ is negligible.

## 3 A Primer on Auction Theory

An auction is a mechanism which runs with some pre-determined rules to sell some item of value. It involves the participation of several parties whose roles are well defined. In the simplest of settings, there is a seller who puts an item on sale and more than one interested buyers compete with each other by placing bids, or the cost they are willing to pay for the item. The highest bidder is announced as the winner and is required to pay a certain amount of money and the item is awarded to this winning buyer. Here we give a brief overview of some of the basic concepts of auction theory.

**Valuation.** Players' valuations define the economic value of an object that is on sale during an auction. It may be the same across the participants in the auction or can be personalised depending on the "value" of the object to each one of them. The valuation is denoted by a function $v(\cdot)$ that takes the object and other observable information that might be specific and personalised to each participant as input and returns the value as a real number $v^* \in \mathbb{R}^+$ (up to some fixed precision). For simplicity, we will refer to the valuation of player $i$ as $v_i$.

**Cost.** The cost defines the economic price that a participant in the auction pays depending on the outcome of the auction. It is denoted by a function $c(b)$ that takes as input a bid $b$ and returns the cost as a real number $c^* \in \mathbb{R}^+$. We assume that the cost function is monotonously increasing with $b$.

**Auction Model.** An auction model describes the set of participants (bidders and sellers), the set of items up for sale and the rules regarding these items, and finally the value of each item for each bidder. The value of an item for each bidder is determined by the bidder's capabilities, preferences, information, and beliefs or what can be collectively called as the *type* of each bidder.

The model accounts for a *mechanism* and an *environment*. A mechanism consists of rules that govern what the participants are permitted to do and how these permitted actions determine outcomes. In this context, an environment comprises of the following: A list of the participants or potential participants, another of the possible outcomes, and another of the bidders' possible *types*.

We consider a set of potential bidders $B_I$ where $I = \{1, 2, \ldots, n\}$. We assume that the types of each bidder are independently and identically distributed (i.i.d.), meaning that the types of each bidder are independent from one another while being from the same distribution. Finally, the utility of bidder $B_i$ is characterised by a function $u_i$ that depends on the bidder's *type* and on the outcome of the auction.

## 3.1 Waiting-time Auction

We first consider the *mediated* setting where an auction is conducted by a trusted auctioneer $\mathbb{A}$ and a set of $n$ bidders $(B_1, \ldots, B_n)$. The auctioneer $\mathbb{A}$ is entrusted with collecting bids from the bidders and awarding the reward to the winner. Moreover, after the bidding phase is over the auctioneer $\mathbb{A}$ reveals the bids of all bidders.

We assume the time to be divided into discrete units which are known to all participants of the auction and to the auctioneer. The auction has several fixed parameters which we assume to be known to every participant:

- The auction good $R$ of some economic value.
- A fixed *token of participation* $Q$ in some arbitrary currency.
- The duration of each auction phase.
- The number of auction rounds.

The auction is composed of three phases, which we describe below.

1. *Bidding Phase:* In the bidding phase each bidder $B_i$ sends its bid $b_i$ along with the token of participation $Q$ to the auctioneer $\mathbb{A}$ through a confidential channel. After a fixed amount of time, $\mathbb{A}$ announces the end of the bidding phase.
2. *Opening Phase:* Let $(b_1, \ldots, b_n)$ be the bids collected in the bidding phase of the same round, let $b_{\mathsf{max}} = \max(b_1, \ldots, b_n)$. In case of *ties* $b_{\mathsf{max}}$ is chosen according to some deterministic order[1]. We denote by $B_{\mathsf{max}}$ the bidder who sent the bid $b_{\mathsf{max}}$. For all $i \in \{1, \ldots n\} \setminus \mathsf{max}$, the auctioneer $\mathbb{A}$ sends $Q$ to $B_i$, whereas $\mathbb{A}$ sends $(Q, R)$ to $B_{\mathsf{max}}$ after $b_{\mathsf{max}}$-many units of time.
3. *Winner Announcement:* $\mathbb{A}$ publicly announces the identity of the winner $B_{\mathsf{max}}$, the amount $b_{\mathsf{max}}$ and *all other bids*.

**Bayesian Nash Equilibrium.** A recent result of Leme et al. [LST12] shows that sequential first-price auctions admit a subgame-perfect Nash equilibrium: This means that there exists a profile of bidding which is a Nash equilibrium in the single round case and, if we arbitrarily fix the outcomes of $\ell$ rounds, the profile also remains a Nash equilibrium for the induced game. Greenwald et al. [GLS12] present a dynamic programming based algorithm to calculate the approximation of such an algorithm. The only difference between our setting and the standard first-price auction is that the winning bidder does not pay directly her bid but has to wait time proportionate to it. If one views the cost of keeping some funds/investment locked for a certain time as the payment (also known as collateral cost), then our waiting-time auction can be cast in the more generic framework of first-price auctions and the existence of a Nash equilibrium follows from the following theorem.

[1]E.g., lexicographical in the commitments of the bidders.

**Theorem 1** ([LST12]). *Sequential first-price auction when a single item is auctioned in each round (assuming that after each round the bids of each agent become common knowledge) has a subgame-perfect equilibrium that does not use dominated strategies, and in which bids in each node of the game tree depend only on who got the item in the previous rounds.*

## 4 Minting Mechanisms and Analysis

In this section we describe the basic minting for PoS systems and we show that with such a mechanism in place, rational users are always incentivised to *hoard* their stake. Later, in contrast to PoS minting, we show that our minting mechanism greatly mitigates this stake hoarding phenomenon.

### 4.1 Utility-Preserving Allocation

To analyse the behaviour of minting mechanisms in relation to stake hoarding we introduce the concept of *utility-preserving stake allocation*, that is similar in spirits to the concept of *Pareto efficiency*[2] [MCWG95]. Analogously to Pareto efficiency, we consider utility functions which assign utilities or benefits to stake allocations. Informally, a utility-preserving stake allocation (or distribution) is an allocation that allows a transition to a different stake allocation where no user decreases his own utility in the process. With this new concept in hand, it becomes possible to analyse if a particular distribution of stakes allows users to trade coins within the system and still maintain their utilities. We give a formal definition below.

**Definition 1** (Utility-Preserving Transition). *Consider two stake allocations $s = (s_1, \dots, s_n)$ and $s' = (s'_1, \dots, s'_n)$ with $\sum_i s_i = \sum_i s'_i = t$. We say a transition from $s$ to $s'$ is utility-preserving, if it holds for all $i \in [n]$ that $u_i(s'_i) \geq u_i(s_i)$.*

**Vanilla PoS Minting.** In PoS systems, the stakeholders assume the role of consensus leaders and propose new blocks to extend the blockchain. These systems ensure that a stakeholder is chosen as the slot leader with probability proportional to one's stake. As an incentive to propose a new block, the consensus leader collects fees from the transactions within the block. As the basic minting mechanism for PoS, we consider the scenario where the consensus leader is also allowed to mint new coins, much similar to what happens in PoW systems (e.g., Bitcoin).

Specifically, consider a proof of stake system where a reward $R$ is given to the consensus leader. Player $i$ becomes consensus-leader with probability $s_i/t$. Let $X_i$ be a random variable which is 1 if player $i$ is consensus leader and 0 otherwise, i.e. the payoff of player $i$ is given by $R \cdot X_i$. Consequently, it holds that $E[R \cdot X_i] = R \cdot E[X_i] = R \cdot \Pr[X_i = 1] = R \cdot \frac{s_i}{t}$, i.e. we define $u_i(s_i) = R \cdot \frac{s_i}{t}$.

In such a system, no non-trivial transition between two stake allocations is utility-preserving. This is shown by the following theorem.

**Theorem 2.** *let $s = (s_1, \dots, s_n)$ and $s' = (s'_1, \dots, s'_n)$ be stake allocations with $\sum_i s_i = \sum_i s'_i = t$ and $s \neq s'$. Then there exists a player $i^*$ for which it holds that $u_{i^*}(s'_{i^*}) < u_{i^*}(s_{i^*})$.*

*Proof.* As $s \neq s'$, there must exists a $j$ with $s_j \neq s'_j$. If $s'_j < s_j$ we set $i^* = j$ and it follows immediately that $u_{i^*}(s'_{i^*}) = R \cdot s'_{i^*}/t < R \cdot s_{i^*}/t = u_{i^*}(s_{i^*})$. On the other hand, if $s'_j > s_j$, there must be a $k$ with $s'_k < s_k$, as otherwise $\sum_i s'_i > \sum_i s_i = t$. In this case, set $i^* = k$ and the statement follows analogously. □

[2]Pareto efficiency is a common notion in game and economic theory used to determine if a particular allocation of resources within a set of players is optimal or not.

**Waiting-Time Auction Minting.** In our proposal, minting is performed via a waiting time auction. Let $X_j^i$ be a random variable which is 1 if player $i$ wins in round $j$ and 0 otherwise. Thus, the payoff of player $i$ is $R \cdot \sum_{j=1}^{\ell} X_j^i$. We will assume that given that player $i$ participates in the auction, his valuation, and therefore his probability of winning does not depend on the stake distribution. I.e. we can write $E[X_j^i] = p_j^i$ for $p_j^i$ that do not depend on $s$. Therefore, it holds that $E[R \cdot \sum_{j=1}^{\ell} X_j^i] = R \cdot \sum_{j=1}^{\ell} p_j^i$ and we can set $u_i(s_i) = R \cdot \sum_{j=1}^{\ell} p_j^i$.

In such a system, every transition of stake-allocations from $s$ to $s'$ for which it holds for all $i \in [n]$ that $s_i, s'_i \geq Q$ is utility-preserving.

**Theorem 3.** *Let $s = (s_1, \ldots, s_n)$ and $s' = (s'_1, \ldots, s'_n)$ be stake allocations with $\sum_i s_i = \sum_i s'_i = t$. If it holds for all $i \in [n]$ that $s_i, s'_i \geq Q$, then it holds for all $i \in [n]$ that $u_i(s'_i) = u_i(s_i)$.*

*Proof.* As it holds for each $i \in [n]$ that that $s_i, s'_i \geq Q$, every player $i$ can participate in the waiting-time auction bid according to their valuation, which is independent of $s$ or $s'$ respectively. The winner of the auction is therefore the same, regardless of whether the stake allocation is $s$ or $s'$. Consequently, the utilities are the same for $s$ and $s'$. $\square$

**Interpreting the Results.** Theorem 2 says that any distribution of stakes within a PoS system with the basic minting strategy will inevitably incentivise the hoarding of stakes, as trading coins will reduce the probability of receiving the newly minted coins. Therefore, users that circulate their coins within the system (i.e., decrease their stake) will be *losing* utility.

In contrast, Theorem 3 says that our minting protocol based on waiting-time auctions mitigates the problem of hoarding; in fact, for each auction round a user is only incentivised to keep a stake of the size of a single participation token. In that case, the user can participate in the auction round, and the probability of winning the newly minted coins will be strictly based on the user's own valuation. The remaining of the stake can be traded into the system without reducing the user's utility.

As an example, consider a system with $t = 100$ total coins, and a user with stake $s = 30$. In case of PoS based minting, to optimise his utility, the user holds his stake throughout the period of the system. In case of our minting, the user needs only $Q = 2$ coins to participate and obtain the newly minted coins. After participating and winning $\ell$ rounds, the user only has locked $\ell \cdot Q$ amount of coins. He can freely trade the rest of the stake for his day-to-day usage. For a pictorial representation we refer the reader to Figure 2. The dotted line represents holding the entire stake and the bars represent locking of participation tokens after winning $\ell$ successive rounds of the auction. The space between the line and the bars (i.e., the gray region) represents the freely tradable stake.

## 4.2 Hoarding in PoS

We present an empirical analysis (in Table 1) that suggests that PoS systems might incentivise their users to hoard their coins, thereby affecting the overall circulation within the system. For comparison, we take into account the PoW based systems Ethereum, Monero, and Zcash, and we compare them with the PoS based systems Cardano, Reddcoin and Peercoin. We consider the total number of transactions per month in each system for the last 6 months starting with May 2018 (data collected from [chab, chaa], where the market capitalization considered is of October 1st, 2018).

Our rationale behind considering these systems is: Ethereum and Cardano are both focusing on smart contract and thus similar on the application level. Monero has a comparable market cap [mar] to Cardano, which is the most popular PoS based system at this time; Reddcoin and Peercoin have existed for approximately the same amount of time as Monero; both Cardano [KRDO17] and Zcash [BCG+14] are based on academic works.

|  |  |  |  | **Total # Tx** |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Currency** | **Consensus** | **Market cap** | **Origin** | **05/2018** | **06/2018** | **07/2018** | **08/2018** | **09/2018** | **10/2018** | **Avg. # Tx/day** | **Avg./Peak** |
| Ethereum | PoW | $23, 6 B | 04/2013 | 25.1 M | 22.5 M | 19.9 M | 19.8 M | 16.1 M | 17.1 M | 654.7 K | 48.5% |
| Monero | PoW | $1.9 B | 05/2014 | 166.4 K | 131.3 K | 124.4 K | 116.2 K | 129 K | 117.4 K | 4.26 K | 39.4% |
| Zcash | PoW | $638 M | 10/2016 | 244.8 K | 254.5 K | 175.9 K | 96.6 K | 100.3 K | 96.7 K | 5.26 K | 31.6% |
| Cardano | PoS [KRDO17] | $2.2 B | 10/2017 | 62.9 K | 43.8 K | 44.5 K | 38.4 K | 43.5 K | 38.5 K | 1.48 K | 13.7% |
| Reddcoin | PoS [Ren14] | $135 M | 02/2014 | 80.7 K | 74 K | 60.6 K | 64.4 K | 69 K | 65.7 K | 2.25 K | 9.4 % |
| Peercoin | PoS [KN12] | $26 M | 04/2013 | 11.4 K | 9 K | 13.6 K | 13.9 K | 9.4 K | 8.7 K | 359 | 4.1 % |

Table 1: Number of transactions per day in cryptocurrency systems based on PoW and PoS.

We can infer from Table 1 that the volume of transactions per day is significantly lower in PoS based systems when compared against PoW based systems. To normalise the difference in transaction volume, we also report the average number of transactions per day, divided by the peak. Here the peak is the number of transactions in the day at which the most transactions occurred in the entire history of the chain (i.e., not only within the last 6 months). As an example, for Ethereum (48.5%) this means that the average number of transactions per day is roughly half compared to the day with most transactions. For all PoS based systems, this measure is below 15%, what again suggests that users tend to hoard after buying the tokens.

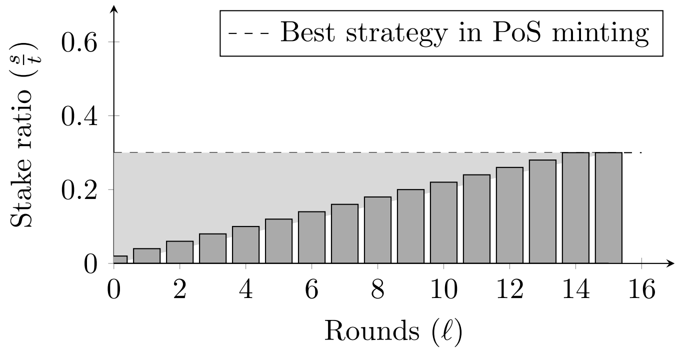

Figure 2: The plot shows the best strategy of a user who wishes to maximise his chance of obtaining the newly minted coins. We consider a system with total number of coins $t = 100$, a user with 30 coins as his stake (i.e., stake ratio 0.3), and participation token $Q = 2$ coins.

## 5 Our protocol

Our minting mechanism implements a first-price waiting-time auction on top of a blockchain protocol. We give a description of the underlying blockchain next.

### 5.1 Blockchain System

In this section we detail the relevant aspects of the underlying blockchain system that is required for our protocol. We treat the blockchain as a public transaction ledger, containing a set of transactions in each block. We consider time to be divided into standard discrete units, such as seconds, minutes, etc. A well defined continuous amount of these units is called a *slot*. Each slot $sl_l$ is indexed for $l \in \{1, 2, 3, \ldots\}$. We assume that users have a synchronised clock that indicates the current time down to the smallest discrete unit. The users execute a distributed protocol to generate a new block in each slot. We assume the slots' real time window properties as in [KRDO17]. In [GKL15, PSS17, KRDO17] it is shown that a "healthy" blockchain must satisfy the properties of *persistence* and *liveness*, which intuitively guarantee that after some time period, all honest users of the system will have a consistent view of the chain, and transactions posted by honest users will eventually be included. We informally discuss the two properties next.

Persistence: Once a user in the system announces a particular transaction as *stable*, all of the remaining users when queried will either report the transaction in the same position in the ledger or will not report any other conflicting transaction as stable. A system parameter $k$ determines the number of blocks that stabilise a transaction. That is, a transaction is stable if the block containing it has at least $k$ blocks following it in the blockchain. We only consider a transaction to be in the chain after it becomes stable.

Liveness: If all the honest users in the system attempt to include a certain transaction into their ledger, then after the passing of time corresponding to $u$ slots which represents the *transaction confirmation time*, all users, when queried and responding honestly, will report the transaction as being stable.

We consider an abstraction of the blockchain protocol, denoted by $\Gamma$, where nodes receive inputs from the environment $\mathcal{Z}$, and interact among each other to agree on an ordered ledger of transactions that achieves persistence and liveness. The blockchain protocol $\Gamma$ is characterised by a set of global parameters and validity conditions which are available to all nodes in the network. The protocol $\Gamma$ provides the nodes with the following set of interfaces which are assumed to have complete access to the network and its users.

- $\{\mathcal{CH}', \perp\} \leftarrow \Gamma.\mathsf{getChain}$: returns a longer $\mathcal{CH}$ in the network if it exists, otherwise returns $\perp$.
- $\{0, 1\} \leftarrow \Gamma.\mathsf{isChainValid}(\mathcal{CH})$: The validity checking takes as input a chain $\mathcal{CH}$ and returns 1 iff the chain satisfies a (public) set of conditions.
- $\Gamma.\mathsf{postTx}(\mathtt{TxType}, dt)$: takes as input the transaction type information and the transaction data. It then constructs a transaction of type $\mathtt{TxType}$ with data $dt$, validate the transaction and include it in the next block.
- $\{\mathtt{txID}, \perp\} \leftarrow \Gamma.\mathsf{isTxStable}(\mathcal{CH}, dt)$: takes as input a chain $\mathcal{CH}$ and some transaction data $dt$ and checks if the transaction containing $dt$ is *stabilised* (w.r.t. the persistence property) in $\mathcal{CH}$. If yes, then it returns the transaction id $\mathtt{txID}$ within $\Gamma$, otherwise it returns $\perp$.
- $\Gamma.\mathsf{broadcast}(dt)$: takes as input some data $dt$ and broadcast it to all the nodes of the system.

The nodes in the $\Gamma$ protocol network have their own local chain $\mathcal{CH}$ which are initialised with a common genesis block. The genesis block contains the information about the addresses of nodes and the spendable balances in each of them. The consensus in $\Gamma$ guarantees the properties of Common prefix, Chain Quality and Chain growth, which in turn gives the properties of persistence and liveness [GKL15, PSS17].

## 5.2 Unlinkable Transactions

In addition to the standard blockchain functionalities, we consider the following interface.

- $\Gamma.\mathsf{postUnlinkTx}(\mathtt{TxType}, dt)$: takes as input the transaction type information $\mathtt{TxType}$ and the transaction data $dt$ and includes the corresponding *unlinkable* transaction in the next block.

Such an interface is used in our main protocol to "shuffle" the addresses of the bidders of the auction. The reason behind this is that an adversary may infer information about the current bid by linking bid transactions from one round to another. This may lead to bid suppression attacks, where the adversary is only willing to include the bids from users that consistently bid lower than the adversary itself, and suppress all other bids. Unlinkable transactions break the link between addresses across several rounds of the auction, thus preventing this selective suppression attack.

Unlinkable transactions have been extensively studied in the literature [BNM+14, RMSK16, RMSK14, MGGR13] and several efficient systems have been proposed. On the one hand we have decentralized mixing protocols [RMSK16, RMSK14, MGGR13] which work on top of an existing (non-unlinkable) blockchain. On the other hand, some blockchain systems have an integrated mechanism to anonymise every transaction and are successfully deployed for real-life applications (cryptocurrencies), such as ZCash or Monero.

Any of the above mentioned proposals is suitable for our system, since we require a very weak notion of unlinkability: Transactions shall be indistinguishable only among the set of payments characterised by the same amount of coins. That is, the amount being transacted is not required to be hidden. The exact property is formalised in Appendix A

## 5.3 Protocol Description

In the following we define the public parameters of the system, the set of validity conditions, and the formal specifications of our construction.

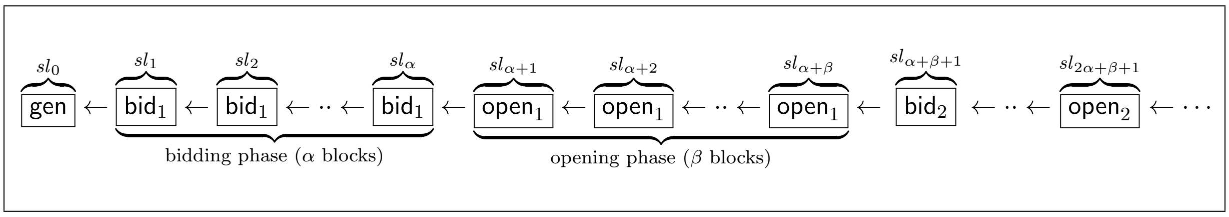

Figure 3: Diagram of the auction phases for each block in the blockchain. The bidding phase of an auction round begins immediately after the opening phase of the previous auction round ends.

### 5.3.1 Global Parameters

Our protocol runs on top of a blockchain system $\Gamma$, and consists of discrete auction rounds $j = (1, 2, \ldots)$. Each auction round consists of two phases: A bidding phase and an opening phase. The bidding phase spans over a sequence of $\alpha$ blocks whereas the opening phase spans over $\beta$ blocks (see Figure 3 for a pictorial description). The parameters $\alpha$ and $\beta$ are fixed throughout the execution of the system. Consider the following NP-language

$$\mathcal{L}_{\mathsf{win}} = \left\{ \begin{array}{l} (\mathtt{crs}_{\mathsf{com}}, \{com_i, \mathsf{addr}_i\}_{i\in[1,\ell]}, (bid^\star, \mathsf{addr}^\star)) : \\ \exists(\{bid_i, r_i\}_{i\in[1,\ell]}) \text{ s.t.} \\ \{com_i = \mathsf{Commit}(\mathtt{crs}_{\mathsf{com}}, \mathsf{addr}_i, bid_i; r_i)\}_{i\in[1,\ell]} \text{ and} \\ (bid^\star, i^\star) = \mathsf{max}\left(\{bid_i\}_{i\in[1,\ell]}\right) \text{ and} \\ \mathsf{addr}^\star = \mathsf{addr}_{i^\star} \end{array} \right\},$$

where the function $(bid^\star, i^\star) \leftarrow \mathsf{max}\left(\{bid_i\}_{i\in[1,\ell]}\right)$ takes as input an ordered set of real numbers and returns the greatest number together with its index. If the output index is not unique, the function selects one deterministically according to some ordering (e.g., lexicographically). Let

$(\mathsf{crsGen}_{\mathsf{win}}, \mathsf{P}_{\mathsf{win}}, \mathsf{V}_{\mathsf{win}})$ be a SNARG system for the language $\mathcal{L}_{\mathsf{win}}$. The global system parameters

$$params = \left(\alpha, \beta, \mathbf{T}, Q, R, \begin{array}{c} \mathtt{crs}_{\mathsf{win}} \leftarrow \mathsf{crsGen}_{\mathsf{win}}(1^\lambda), \\ \mathtt{crs}_{\mathsf{com}} \leftarrow \mathsf{Setup}(1^\lambda) \end{array}\right)$$

consist of the auction parameters $(\alpha, \beta)$, the hardness parameter $\mathbf{T}$ (of the time-lock puzzle refer subsection 2.1), a token value $Q$, a reward value $R$, and a set of common reference strings.

### 5.3.2 Chain Validity.

In the following we describe the conditions that determine the validity of a chain in our system. The interface $\mathsf{isChainValid}(\mathcal{CH}')$ takes as input a chain $\mathcal{CH}'$ and validates all transactions in the chain according to the rules that will be described below. It returns 1 if and only if all of the transactions are valid. Users of the blockchain are indexed by addresses $\mathtt{addr}$, which belong to a certain efficiently samplable domain $\mathbb{A}$ (note that a node in the network may be associated with multiple addresses). We define the balance function $\mathsf{balance}(\mathcal{CH}, \mathtt{addr})$ that takes as input the chain $\mathcal{CH}$ and an address $\mathtt{addr}$ and returns the *spendable balance* associated with $\mathtt{addr}$. The spendable balance is initially 0 for all addresses and it is modified by different types of transactions. We now define the different types of transactions and describe how to validate each of them.

Spend transactions: Standard spend transactions $\mathtt{payTx}$ move coins between addresses and are of the form $(\mathtt{addr}_0, \mathtt{addr}_1, val)$. Let $\mathcal{CH}$ and $\mathcal{CH}'$ be the state of the chain before and after this transaction, respectively. A spend transaction is considered *valid* if and only if

$$\mathsf{balance}(\mathcal{CH}, \mathtt{addr}_0) \geq val.$$

Such a transaction updates the spendable balance of the two addresses as

$$\mathsf{balance}(\mathcal{CH}', \mathtt{addr}_0) = \mathsf{balance}(\mathcal{CH}, \mathtt{addr}_0) - val$$

and

$$\mathsf{balance}(\mathcal{CH}', \mathtt{addr}_1) = \mathsf{balance}(\mathcal{CH}, \mathtt{addr}_1) + val.$$

Bid transactions: Bid transactions $\mathtt{bidTx}$ are used to post bids in each auction round and are of the form $(com, tlp, \mathtt{addr}, j)$. Bid transactions are considered *valid* and accepted into the chain if and only if the round $j$ is the current and $\mathsf{balance}(\mathcal{CH}, \mathtt{addr}_0) \geq Q$. Let $\mathcal{CH}'$ be the state of the chain right after this transaction, the spendable balance is updated as

$$\mathsf{balance}(\mathcal{CH}', \mathtt{addr}) = \mathsf{balance}(\mathcal{CH}, \mathtt{addr}) - Q.$$

Steal transactions: Steal transactions $\mathtt{stealTx}$ are used to steal the participation token, in case one user posts an *inconsistent bid.* Steal transactions are of the form $(\mathtt{addr}_0, \mathtt{addr}_1, \pi, (m, r), j)$ and are considered valid if the $j$-th round is the current and the $j$-th bidding phase contains an inconsistent bid of the form $(com, tlp, \mathtt{addr}_0, j)$ such that

$$\mathsf{PVer}(tlp, (m, r), \pi) = 1 \text{ and } \mathsf{Commit}(\mathtt{crs}, \mathtt{addr}_0, m; r) \neq com.$$

If the transaction is valid, the spendable balance of $\mathtt{addr}_1$ is updated as follows

$$\mathsf{balance}(\mathcal{CH}', \mathtt{addr}_1) = \mathsf{balance}(\mathcal{CH}, \mathtt{addr}_1) + Q.$$

In case of multiple $\mathtt{stealTx}$ transactions for the same bid, only the first instance is considered valid.

Mint transactions: Mint transactions $\texttt{mintTx}$ are used to introduce new coins in the system and are of the form $(\texttt{addr}^\star, \mathit{bid}^\star, \pi_{\mathsf{win}}, R, j)$. To validate a mint transaction, a node checks if no valid mint transaction for the same round already is included in the blockchain. If not, it fetches all the bid transactions in the bidding phase of the corresponding auction round to retrieve the vector $\{com_i, \texttt{addr}_i\}_{i\in[1,\ell]}$. Addresses for which a valid steal transaction is present in the current round (i.e., they published a malformed bid) are excluded from this set. A mint transaction is *valid* if and only if

$$\mathsf{V}_{\mathsf{win}}\left(\texttt{crs}_{\mathsf{win}}, \left(\begin{array}{l}\texttt{crs}_{\mathsf{com}},\\ \{com_i, \texttt{addr}_i\}_{i\in[1,\ell]}\\ (\mathit{bid}^\star, \texttt{addr}^\star)\end{array}\right), \pi_{\mathsf{win}}\right) = 1$$

and the round $j$ is the current. The balance of all non-winning accounts is immediately increased by $Q$. Let $\mathcal{CH}'$ be the state of the chain *after* $\mathit{bid}^\star$-many units of time. The value of $\mathsf{balance}(\mathcal{CH}', \texttt{addr}^\star)$ is increased by $R+Q$. That is, the newly minted coins are *not spendable* until $\mathit{bid}^\star$-many units of time are elapsed.

### 5.3.3 The Minting protocol

Our protocol generates new coins periodically, over fixed time intervals. The complete specification of our protocol is given in Figure 4. The coin generation mechanism spans over a round of a first-price waiting-time auction. At the beginning of each time slot, users in the network receive and update their local chain and generate a fresh and unlinkable bidding address.

During the bidding phase (step 1), a user obtains a bid as input from the environment and commits to the bid (provided she has sufficient balance to afford the participation token). Along with the commitment, she also generates a time-lock puzzle of the unveil information of the commitment. This time-lock ensures that after a specified time the unveil information can be recovered if a user fails to reveal the bid herself (e.g. the user went offline). She then posts a transaction containing the commitment and the time-lock puzzle.

During the opening phase (step 2), users that posted a bid in the bidding phase can broadcast to the network the unveil information of their commitment, so their bids can be opened. If the opening phase is over and no winner was announced yet (step 3), the miners determine the winner by opening all the bids posted during the bidding phase. If there is no unveil information available for a commitment (i.e., the bid is unknown), then the miner solves the time-lock puzzle for that commitment and recovers the unveil information. If the puzzle does not contain a well-formed unveil, the miner posts the corresponding recovery proof, thereby stealing the token $Q$ from the faulty address. Such a bidding transaction is ignored in the subsequent computations for deciding the winner of the auction round.

Once all openings are computed, the miner includes in its block a mint transaction that assigns newly minted coins to the winner of the auction. The new coins are only redeemable after the time corresponding to the winning bid has elapsed, and a SNARG proof guarantees that every bid posted in the bidding phase was considered in the auction round. The veracity of such a statement can be verified using minimal space over the blockchain, since the SNARG proof is succinct.

## 5.4 Protocol Analysis

The following theorem shows that our implementation preserves the subgame-perfect Nash-equilibria of the mediated game. In other words, we formally argue that our protocol implements a waiting-time first-price auction on top of a blockchain (with its own set of incentives). Note

**(Fetch chain).** At the beginning of each time slot $sl_l$, for $l \in \mathbb{N}$, each node attempts to update its local view by calling $\mathcal{CH} \leftarrow \Gamma.\mathsf{getChain}$. If $\mathsf{isChainValid}(\mathcal{CH}) = 1$ then the node sets $\mathcal{CH}$ as the new local chain.

**(Address Generation).** Starting from an address $\mathtt{addr}$ such that $\mathsf{balance}(\mathcal{CH}, \mathtt{addr}) \geq Q$, the node generates a fresh bidding address $\mathsf{addr}_B$ and posts an unlinkable transaction through $\Gamma.\mathsf{postUnlinkTx}(\mathtt{payTx}, (\mathtt{addr}, \mathsf{addr}_B, Q))$.

**(Auction round)** At the beginning of an auction round $j$, all the nodes start with a bidding address $\mathsf{addr}_B$. Each node checks the local chain $\mathcal{CH}$ to determine the current phase (bidding, opening or winner announcement) and proceed as follows.

1. **(Bidding phase)**
   (a) Receive input $bid$ from the environment $\mathcal{Z}$
   (b) If the bid-transaction has not yet been posted in the current phase yet, then compute $com \leftarrow \mathsf{Commit}(\mathtt{crs}_{\mathsf{com}}, \mathsf{addr}_B, bid; r)$, using some random coins $r$, to commit to $bid$
   (c) Create a time-lock puzzle $tlp$ encapsulating the unveil information of $com$ by running $tlp \leftarrow \mathsf{PGen}(1^\lambda, \mathbf{T}, (bid, r); r')$, using some random coins $r'$
   (d) Post a bid transaction through $\Gamma.\mathsf{postTx}(\mathtt{bidTx}, (com, tlp, \mathsf{addr}_B, j))$
2. **(Opening phase)**
   (a) Check the stability of the bid transaction by verifying that $\Gamma.\mathsf{isTxStable}(\mathcal{CH},$ $(com, tlp, \pi_{\mathsf{bid}}, j, \mathsf{addr}_B)) \neq \perp$. If the transaction is stable, then broadcast the unveil information through $\Gamma.\mathsf{broadcast}(\mathsf{addr}_B, bid, r)$.
3. **(Winner announcement)**
   (a) If a valid winner announcement transaction for the current round already exists, skip the steps below
   (b) Collect all valid openings that are broadcasted for the current auction round and determine the corresponding bids and addresses
   (c) For each of the unopened bids $(com_i, tlp_i, \mathsf{addr}_{B_i})$ solve the corresponding time-lock puzzle $tlp_i$ by computing $((bid_i, r_i), \pi_{tlp}) \leftarrow \mathsf{PSolve}(tlp_i)$. If $\mathsf{Commit}(\mathtt{crs}_{com}, \mathsf{addr}_{B_i}, bid_i; r_i) \neq 1$ then post the steal transaction $\Gamma.\mathsf{postTx}(\mathtt{stealTx}, (\mathsf{addr}_{B_i}, \mathtt{addr}, \pi_{tlp}, (bid_i, r_i), j))$, where $\mathtt{addr}$ is the address of the miner.
   (d) After this step, a complete list of all bids together with the corresponding random coins and addresses of the bidders is available $\{bid_i, r_i, \mathsf{addr}_{B_i}\}_{i\in[1,\ell]}$
   (e) Determine the highest bid $bid^\star$ and the corresponding address $\mathsf{addr}_B^\star$ by $(bid^\star, i^\star) \leftarrow \mathsf{max}\left(\{bid_i\}_{i\in[\ell]}\right)$ and set $\mathsf{addr}_B^\star = \mathsf{addr}_{B_{i^\star}}$
   (f) Run
   $\pi_{\mathsf{win}} \leftarrow \mathsf{P}_{\mathsf{win}}\left(\mathtt{crs}_{\mathsf{win}}, \left(\mathtt{crs}_{\mathsf{com}}, \{com_i, \mathtt{addr}_i\}_{i\in[1,\ell]}, (bid^\star, \mathtt{addr}^\star)\right), \left(\{bid_i, r_i\}_{i\in[1,\ell]}\right)\right)$
   to generate a proof that $\mathsf{addr}_B^\star$ is the highest bidder among all $\ell$ bids and the highest bid value is $bid^\star$
   (g) Post the minting transaction through $\Gamma.\mathsf{postTx}(\mathtt{mintTx}, (\mathsf{addr}_B^\star, bid^\star, \pi_{\mathsf{win}}, R, j))$

Figure 4: Waiting-time auction based minting protocol

that our analysis holds under the condition that $R \leq m \cdot F$, where $F$ is the value of a transaction fee, $m$ is the number of players in the auction, and $R$ is the value of the reward (i.e., newly minted coins). This ensures that it is more profitable for a miner to include all bids (thereby collecting fees) rather than dropping even one anonymous bid to increase its own odds in the auction. Therefore, all bids will eventually be posted in the blockchain. In practice, $R$ can be adjusted to meet this bound with a conservative estimation on the number of active players in the auction. The analysis is deferred to Appendix B.

**Theorem 4** (Subgame-perfect Nash-equilibria)**.** *Let $m$ be the number of bidders in the auction, $F$ be the transaction fee for each bid, and $R$ be the reward. If $R \leq m \cdot F$ then the protocol in Figure 4 implements a sequential mediated waiting-time auction.*

## 5.5 System Parameters

In this section we discuss about plausible settings for the parameters of our system.

**Minting Reward ($R$).** In our system, each transaction is associated with a fee $F$, that the user offers to the miner as an incentive to include her transaction. This holds also for bid transactions. To avoid selective suppression attacks, where the miner does not include a transaction to increase her odds to win the auction, we need to make sure that (in the long run) it is more beneficial to include all bids and thereby collect all transaction fees. As discussed above, our analysis requires that $R \leq m \cdot F$, where $m$ is the number of potential bidders in a round of the protocol. Note that $m$ denotes the number of participants in the auction, among which some might be "blocked" by the miners. The value of $m$ can be estimated from the addresses present in the blockchain and from the previous runs of the protocol. The value of $R$ is set accordingly.

**Participation Token ($Q$).** In our system, every address (user) can participate in the auction as long as it holds at least $Q$ coins. To claim the reward, the winning bidder is required to "freeze" these $Q$ coins until the amount of time corresponding to her bid. Ideally, $Q$ should correspond to an amount of coins that is meaningful enough so that players are not willing to freeze it for an absurd amount of time. The upper bound guarantees that most users of the system can compete for the reward, whereas the lower bound avoids malicious attacks where players prevent the creation of new coins at very little cost (e.g., by bidding an unreasonable amount of time).

**Difficulty Parameter ($\mathbf{T}$).** The difficulty parameter $\mathbf{T}$ of time-lock puzzles depends on the advancements of microchips design and manufacturing. An estimate of the performance of the best processor that is currently available can be made and the difficulty parameter can be calibrated by the network itself in periodic intervals. For our analysis, we will assume that the time-lock puzzle has a fixed gap $e < 1$ and that the hardness parameter $\mathbf{T}$ is chosen in such a way that $\mathbf{T}^e$ is longer than the bidding-phase.

**Auction Phases ($\alpha$ and $\beta$).** In our protocol, the opening phase begins immediately after the last block in the bidding phase (of that auction round) is published. Recall that a transaction is required to be present in the blockchain for at least $k$ blocks (persistence) in order to be considered stable. It is therefore precautious to *not* broadcast the unveil information before one's bid is stabilised. For this reason we require that the opening phase ($\beta$) spans over an amount of blocks strictly larger than $k$ (for eg., $k = 6$ in case of Bitcoin). On the other hand, the bidding phase ($\alpha$) should not be as long as to allow some resourceful user to solve some puzzle (and thus learn some bid) before the phase is over. This prevents players from adaptively choosing bids according to other players' bids. Therefore, we require that $\alpha < \mathbf{T}$. Under this constraint, the bidding phase would be over by the time anyone solves a time-lock puzzle.

### 5.6 Bootstrapping the System

The simplest way to bootstrap our system is to assign coins to an initial set of users and start the auction rounds thereafter. One can also apply our minting mechanism on top of any existing currency system where funds are already distributed across users. This can be done by taking a snapshot of the existing currency system with its existing keys and balances and then start our minting mechanism on top of the existing stake distribution. Since the system is already running we do not require any additional assumption given that our mechanism is incentive compatible (see Theorem 4). We only require the system to be compatible with coin mixers (Mixing Services, Coinshuffle(++), etc.) to support unlinkable transactions. Note that there is no hard limit on the number of initial participants.

Yet another possibility is to bootstrap the blockchain system with an existent minting protocol (such as proof of work), and switch to our minting protocol after a fixed amount of time. The initial mining process can ensure that new coins are introduced into the system and distributed among users so that, once the minting mechanism is switched, these users possess enough spendable balance to participate in the auction.

## 6 Implementation

In this section we report a python proof-of-concept implementation of our protocol from subsection 5.3. We implement a full-fledged Blockchain system with our minting mechanism in a way that it is completely independent of the underlying consensus used in the system. For this, we rely on the libSNARK library [Lab18] to produce the SNARG proof for new minting transactions. Although libSNARK's proof require a trusted setup, we remark that our system is completely parametric in the SNARG algorithm and we can switch to setup-free alternatives [AHIV17, WTTW, BCC$^+$16, BSBHR17, BBB$^+$17] essentially for free. For convenience, our prototype uses the reference implementation of libSNARK.

Apart from standard Blockchain operations (e.g., coin transfer, public verification, etc.), our system accommodates all of the required auction operations and validation rules specified in our minting protocol. In the next sections we describe the implementation in detail, followed by benchmarking.

### 6.1 Cryptographic Components

Our proof-of-concept blockchain system has been fully implemented on python 3 and it mimics all the basic functionalities of Bitcoin, including a subset of Bitcoin's script language. We rely on a proof-of-work based consensus for conceptual simplicity and we set the average time between block creation to be 10 minutes; the duration of a full auction round extends for 100 blocks, with the bidding phase being 50 blocks. As in Bitcoin, we use the ECDSA signature scheme over the elliptic curve `secp256k1` which has a signature of size 65-bytes, private key of size 32-bytes and public-key of size 65-bytes. We use the fastecdsa [Kue18] library for ECDSA signatures. The two main components of our system are bid and mint transactions, that we describe in the following.

**Bid Transactions.** Like standard transactions, a bid transaction consists of inputs and outputs; the difference here is that we limit a bid transaction to contain a single input which points to a previous output with *at least* the transaction fees plus the participation token (which we set as 10 coins in the prototype). The output of a bid transaction consists of a commitment to a bid (in seconds) and a *tlp* that contains the unveil information of the commitment. The average size for a bid transaction (including input and output) in our prototype is 289 bytes. The

| | | # Tx per block size | | |
|---|---|---|---|---|
| **Transaction Type** | **Size** | 1MB | 8MB | 12MB |
| Bid Tx | 289 bytes | 3.4 K | 27.6 K | 41.5 K |
| Mint Tx | 252 bytes | 3.9 K | 31.7 K | 47.6 K |
| Spend Tx | 165 bytes | 6.0 K | 48.4 K | 72.7 K |
| Unveil Tx | 56 bytes | 17.8 K | 142.8 K | 214.8 K |
| Steal Tx | 2.2 KB | 454 | 3.6 K | 5.4 K |

Table 2: Number of transactions of each type that would fit in a single block. We stress however that the bidding phase can consist of multiple blocks, and that only a single mint transaction is allowed per auction round.

commitment to bids are implemented as SHA-256 commitments computed using the libSNARK SHA-256 hash function, with the inputs being the bid, a 64-bit integer (representing seconds), and the randomness, a 128-bit integer. The unveil information for the commitments are the bid itself and the randomness. To verify the correctness of the unveil, we use it to recompute the hash function and check for equality with the initially committed value.

**Time-Lock Puzzles.** The *tlp* we deploy is based on the work of Pietrzak [Pie18], which leverages repeated squaring as a non-parallelisable operation. We conservatively set the hardness parameter $\mathbf{T}$ to be $2^{35}$, which keeps the *tlp* locked for more than 15 hours with the hardware specified in subsection 6.3. We instantiate the *tlp* with an RSA modulus of 512 bits, which we estimate to be sufficient for hiding a value for less than a day. In Pietrzak's scheme a *tlp* is of the form $(N, x, y)$, where $N$ is an RSA modulus, $x$ is an element of $\mathbb{Z}_N^*$, and $y$ is constructed as $H(x^{\mathbf{T}}) \oplus m$. Given $x^{\mathbf{T}}$ one can efficiently recover $m$. Furthermore, one can efficiently compute a proof $\pi$ that a certain $z = x^{\mathbf{T}}$, which consists of $\log(\mathbf{T}) - 2$ elements of $\mathbb{Z}_N^*$, using the technique described in [Pie18]. We refer the reader to [Pie18] for further details.

**Mint Transactions.** A mint transaction has no inputs and it contains exactly two outputs that we detail next. The first output contains a 137-byte SNARG proof, along with the highest bid (8-bytes), and the commitment to the highest bid (32-bytes), thus adding to a total of 177-bytes. The second output is a *pay-to-pubkey-lock* type transaction, that is a standard *pay-to-pubkey* transaction with a lock-time corresponding to the value of the winning bid; the transaction cannot be redeemed until the lock-time expires. The size of a mint transaction is approximately 252-bytes.

**LibSNARK.** For the SNARG in the mint transactions we use the libSNARK [Lab18] implementation of the system described in [Gro16]. We build a python wrapper around the libSNARK argument system and use it as a shared library. To produce a proof of the auction winner the miner supplies a list of bid commitments and a list of unveils, and the statement is that the miner knows the unveils for all bid commitments and that all bid commitments are smaller than the first bid commitment on the list. Since we use SHA-256 commitments for the bids we can employ the constraint systems already defined by libSNARK. The libSNARK library requires a set-up phase, that runs before the initialisation of the Blockchain system and generates the public parameters. In our prototype we run tests for up to 750 bids in each auction round.

## 6.2 Optimisations

As shown in Figure 5, the time to generate a SNARG can grow significantly with the number of bids. A possible trade-off to allow more bids in a round is to exclude the part of unveils

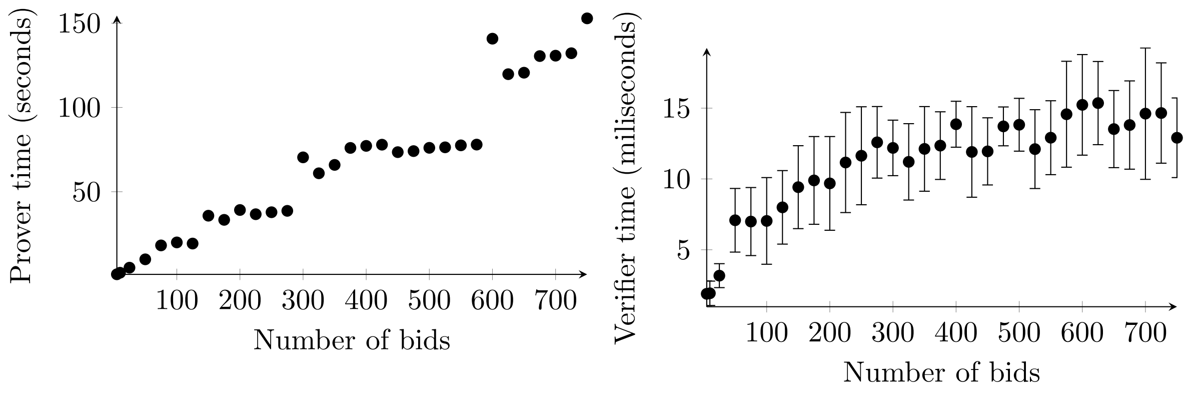

Figure 5: The graphs show the average time to generate/verify a SNARG in a mint transaction. The average is taken over the run of 100 experiments for each parameter value. The error bars display the standard deviation of the measurements.

from the proof and publish them in the Blockchain instead. Unveil transactions contain the bid commitment and its unveil information; their total size sums up to around 56-bytes. That way, the miner creating a payout block can publish the SNARG for part of the bids (as usual) and publish the unveil information of the remaining bid commitments that were not included in the SNARG; the reward is assigned to the highest bidder among the highest bid in the proof and the highest bid with the unveil published. To verify the validity of such a mint transaction a miner opens the bid commitments (with the unveil in the Blockchain) and verify the proof for the remaining bids; if the mint transaction is assigned to the highest bidder among the two sets then the transaction is valid. Taking this time-space tradeoff even further, one can completely *replace* SNARGs if willing to include all the bid unveils in the Blockchain.

We stress that the only reason to compute a SNARG in the mint transaction is its succinctness. Without including any unveil transactions in the Blockchain, and for a reasonable amount of bids (around 750) a proof can be generated in under 3 minutes and verified almost instantly in our current implementation (see Figure 5).

## 6.3 Benchmarking

In this section we detail the performance achieved by our implementation running several experiments. The benchmarking was performed in a virtual environment on a Linux server with the following specifications.

- Intel Xeon Gold 6132 CPU (32 cores) @ 2.60GHz
- 64GB of RAM
- Debian Linux 4.9.0-6-amd64
- Python 3.6.4, fastecdsa 1.6.4, and the latest libSNARK.

We measure the time to generate and to verify SNARG proofs for a mint transaction varying the number of bids considered in each auction round. For each experiment we generate fresh bid commitments and we run 100 iterations of each experiment, taking the average time among all the iterations. The results of the experiments shown in Figure 5 were measured considering the wait time, and with the libSNARK multicore mode enabled (32 cores). The graph on the left of Figure 5 shows outlier points for 300 and 600 bids; this is due to parallelisation.

In Table 2 we show the different types of transactions introduced by our minting mechanism and how many transactions of each type could be included in different sizes of blocks. We stress that Steal transactions happen only in the case of a malformed bid, which does not happen in a rational run of the protocol. A block of size 8MB allows us to accommodate around 1K bid transactions per block still leaving roughly 70% of the block space free for standard spend transactions. Extending the bidding phase to about 10 blocks gives a total of 10K bids in a single auction round, and only taking 30% of the available block space for bid transactions. This gives strong evidences that the system can scale to support a large amount of users.

## 7 Discussion

Due to space constraints, we informally discuss the intuition behind how we prevent some of the common attacks against our minting protocol of subsection 5.3. For a more formal treatment we refer the reader to Theorem 4 and its proof.

**Bid Suppression.** The most straightforward attack for the adversary is to suppress bids from a block during the bidding phase. By suppressing bids from a block, the adversary can increase its chances of winning the newly minted coins. As we show in the analysis of Theorem 4, this strategy has ultimately a *decreasing* payoff, and therefore will be avoided by the rational adversarial miner. The intuition behind this argument is that by suppressing bids, the adversary will be forfeiting the transaction fees incurred by the bid transactions, what would be less profitable than simply including all the bids and following the protocol.

**Denial-of-Coin.** A denial-of-coin attack is when the adversary tries to stop the creation of new coins in the system. One way to achieve this goal is to bid an incredibly high amount of time (way above one's valuation), such that the newly minted coins would remain locked (practically) forever. This is not a profitable attack for the rational adversary, since this strategy would quickly lock all funds of the adversary, eventually reestablishing the coin supply. Furthermore, the attacker must be heavily invested in the currency to launch such an attack and thus he is hurting primarily himself with this manoeuvre.

**Denial-of-Service.** A possible denial-of-service attack is for the adversary to spam the network with many bid transactions in order to stall the network and avoid honest users from participating in the bidding process. Our protocol avoids this by charging a transaction fee for each bid posted. In that way, for the adversary to be able to spam the network he would have to decrease his payoff significantly.

Another vector of attack to slow down the network is to post (well-formed) bids but not their openings. This causes the miners to incur in additional computational efforts to brute-force the time-lock puzzles. This behaviour falls outside of the scope of a rational attacker since it only delays the resolution of the auction and the release of his own participation tokens (and therefore increases his collateral cost). Such a behaviour can also be prevented by penalising the absence of openings with additional fees.

**Mint Suppression.** This attack happens when the miner refuses to include a valid minting transaction into the block being mined. Such an attack is not rational for any miner because at this point of the execution the winner is already determined, although not yet announced. The miner cannot change the winner of the auction and therefore does not gain any advantage by denying to accept the minting transaction.

**Malformed Bids.** An attacker could see posting inconsistent time-lock puzzles as an opportunity to slow down the system, since miners need to solve a time-lock puzzle to eventually realise that the bid is not well-formed. As shown in our analysis in Appendix B, this behaviour is not profitable for any attacker, since any miner who fails to solve a malformed time-lock puzzle can produce a recovery proof and steal the participation token of the bidder.

## 8 Related Work

**Blockchain and Consensus.** Nakamoto [Nak08] proposed Bitcoin, the first currency system with a consensus protocol based on Proof of Work (PoW) which was originally put forth by Dwork and Naor [DN93]. The underlying protocol of Bitcoin was dubbed as the *Blockchain protocol* and a formal analysis of its security definitions and properties can be found in the works of Garay et al. [GKL15, GKL17] and Pass et al. [PSS17]. BitcoinCash, Litecoin (variants of Bitcoin), Zcash and Monero are some of the popular currencies based on PoW. One among several other alternatives proposed was Proof of Stake (PoS) based consensus where a consensus leader proves she holds a stake in the system. The proposal was formally analysed with the assumption of a synchronous [KRDO17] and semi-synchronous network [DGKR18], and in the recent work of Badertscher et al. [BGK$^+$18] which concerns with composability of PoS blockchains. There are several currency systems that are based on different versions of PoS, namely, Cardano (based on Ouroboros), Reddcoin [Ren14], Peercoin [KN12] among possibly many others. Proofs of Space [DFKP15] is another proposal put forth that relies on a prover proving to a verifier that she has sufficient disk space, to achieve a consensus. Spacecoin [PPA$^+$15] is a currency system whose consensus is based on proofs of space.A closely related proposal is Proof of Secure Erasure formalised by Ateniese et al. [ABFG14] and Karvelas, and Kiayias [KK14], where a space restricted prover convinces a verifier that she has erased some size of memory from her storage.

**Blockchain and Minting.** In all of the above mentioned consensus mechanisms, the consensus leader in the blockchain is also the one who receives the incentive in the form of newly minted coins, with the exception of proof of stake. Selfish mining attacks (where a miner mines a block selfishly and later hopes to make his chain longer and accepted) in case of Nakamoto's blockchain protocol were discussed in the bitcoin forum [mtg10] and later analysed and improved by Eyal and Sirer [ES14], Sapirshtein et al. [SSZ16] and Nayak et al. [NKMS16]. Pass and Shi [PS17] designed a new blockchain protocol called Fruitchain that ensures that no coalition that has less than the majority of the computational power can gain more by deviating from the protocol. Concurrently, Carlsten et al. [CKWN16] showed the possible instability in the future of Bitcoin as a result of incentives through transaction fees only.

**Auctions on Blockchains.** Running auctions on blockchains has been gaining attention given its nature of public verifiability [Acc17]. There are several existing proposals for running different variants of auctions. Kosba et al.'s HAWK [KMS$^+$16] employ smart contracts to run auctions on top of a blockchain. They assume the existence of a special entity called *Manager* who is entrusted to run the auction contract. The manager is aware of the bidders' inputs and is trusted to not disclose that information. Strain [BK] aims to decrease the amount of interaction, while relying on a semi-honest *judge* who does not collude with any bidders and produces proof of winner.

## 9 Conclusions and Open Problems

In this work we initialise the study of minting mechanisms in cryptocurrencies as a primitive of independent interest in order to closely model a real world fiat currency. We propose the first minting mechanism that when implemented in a PoS system, mitigates hoarding of tokens by introducing a steady inflation, is completely decoupled from the consensus of the underlying blockchain, and does not rely on a semi-trusted party. Our key technical contribution is a protocol to run an auction over a blockchain that does not require a semi-trusted auctioneer. In future works we hope to investigate the functioning of our rewarding mechanism considering variable bidding fees and an asynchronous setting.

## References

[ABFG14] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In *International Conference on Security and Cryptography for Networks*, pages 538–557. Springer, 2014.

[Acc17] Accenture. How blockchain can bring greater value to procure-to-pay processes. 2017. https://www.accenture.com.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2087–2104. ACM, 2017.

[ass] Bitcoin is an asset, not a currency. https://tinyurl.com/ydc2nnzw.

[BBB$^+$17] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Efficient range proofs for confidential transactions. Technical report, Cryptology ePrint Archive, Report 2017/1066, 2017., 2017.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Fischlin and Coron [FC16], pages 327–357.

[BCG$^+$14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, Berkeley, CA, USA, May 18–21, 2014. IEEE Computer Society Press.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BGJ$^+$16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 345–356. ACM, 2016.

[BGK$^+$18] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 913–930, New York, NY, USA, 2018. ACM.

[BK] Erik-Oliver Blass and Florian Kerschbaum. Strain: A secure auction for blockchains.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[BNM$^+$14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *Lecture Notes in Computer Science*, pages 486–504, Christ Church, Barbados, March 3–7, 2014. Springer, Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BSBHR17] Eli Ben-Sasson, Iddo Bentov, Ynon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Manuscript.(2017). Slides at https://people. eecs. berkeley. edu/˜ alexch/docs/pcpip_bensasson. pdf*, 2017.

[chaa] Bitinfocharts. https://tinyurl.com/yc4pkkjn.

[chab] Coin metric charts. https://coinmetrics.io/api/.

[CKWN16] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 585–605, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[DGKR18] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[DN93] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[FC16] Marc Fischlin and Jean-Sébastien Coron, editors. *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[GKL15] 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, Part II*, volume 9057 of *Lecture*

*Notes in Computer Science*, pages 281–310, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[GLS12] Amy Greenwald, Jiacui Li, and Eric Sodomka. Approximating equilibria in sequential auctions with incomplete information and multi-unit demand. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2321–2329. Curran Associates, Inc., 2012.

[Goo] Dan Goodin. Active attack on tor network tried to decloak users for five months. https://tinyurl.com/y7995tkc.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Fischlin and Coron [FC16], pages 305–326.

[HP15] Joseph Y. Halpern and Rafael Pass. Algorithmic rationality: Game theory with costly computation. *J. Economic Theory*, 156:246–268, 2015.

[Hum07] Jeffrey Rogers Hummel. Death and taxes, including inflation: the public versus economists. *Econ Journal Watch*, 4(1):46, 2007.

[Kin13] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. 2013.

[KK14] Nikolaos P. Karvelas and Aggelos Kiayias. Efficient proofs of secure erasure. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 520–537, 2014.

[KMS$^+$16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012. https://decred.org/research/king2012.pdf.

[Kot] Ivana Kottasova. Bitcoin is not a currency. https://tinyurl.com/yca958rk.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[Kue18] Anton Kueltz. fastecdsa: Python library for fast elliptic curve crypto. https://github.com/AntonKueltz/fastecdsa, 2018.

[Lab18] SCIPR Lab. libsnark: a c++ library for zksnark proofs. https://github.com/scipr-lab/libsnark, 2018.

[LST12] Renato Paes Leme, Vasilis Syrgkanis, and Éva Tardos. Sequential auctions and externalities. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 869–886. Society for Industrial and Applied Mathematics, 2012.

[mar] Coinmarketcap. https://coinmarketcap.com.

[MCWG95] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[Met] Tom Metcalf. The wealthy are hoarding $ 10 billion of bitcoin in bunkers. https://tinyurl.com/yaymn3rd.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[Min17] Mining hardware comparison. 2017. https://tinyurl.com/4pjhy5t.

[MJS$^+$14] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 475–490. IEEE, 2014.

[mtg10] mtgox. 2010. https://tinyurl.com/y9alux2j.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. https://bitcoin.org/bitcoin.pdf.

[NKMS16] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 305–320. IEEE, 2016.

[OM14] KJ O'Dwyer and D Malone. Bitcoin mining and its energy footprint. In *IET Conference Proceedings*. The Institution of Engineering & Technology, 2014.

[OS90] Athanasios Orphanides and Robert M. Solow. Chapter 6 money, inflation and growth. volume 1 of *Handbook of Monetary Economics*, pages 223 – 261. Elsevier, 1990.

[Pec13] Morgen E Peck. The bitcoin arms race is on! *IEEE Spectrum*, 50(6):11–13, 2013.

[Pie18] Krzysztof Pietrzak. Simple verifiable delay functions. Cryptology ePrint Archive, Report 2018/627, 2018. https://eprint.iacr.org/2018/627.

[Poo17] Bitcoin mining pools. 2017. https://tinyurl.com/y8pdk922.

[PPA$^+$15] Sunoo Park, Krzysztof Pietrzak, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. Spacemint: A cryptocurrency based on proofs of space. *IACR Cryptology ePrint Archive 2015:528*, 2015. https://eprint.iacr.org/2015/528.pdf.

[PS17] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 315–324. ACM, 2017.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 643–673, 2017.

[Ren14] Larry Ren. Proof of stake velocity: Building the social currency of the digital age. *Self-published white paper*, 2014.

[RMSK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014.

[RMSK16] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. P2p mixing and unlinkable bitcoin transactions. *IACR Cryptology ePrint Archive*, 2016:824, 2016.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.

[Sat11] Khayroollo Sattarov. Inflation and economic growth. analyzing the threshold level of inflation.: Case study of finland, 1980-2010., 2011.

[SSZ16] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[Tay13] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, page 16. IEEE Press, 2013.

[Tsi89] Sho Chieh Tsiang. A critical note on the optimum supply of money. In *Finance Constraints and the Theory of Money*, pages 331–348. Elsevier, 1989.

[Woo11] Michael Woodford. *Interest and prices: Foundations of a theory of monetary policy.* princeton university press, 2011.

[WTTW] Riad S Wahby, Ioanna Tzialla, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup.

## A Definitions

In this section we present the formal definitions that are going to be used in our analysis.

### A.1 Probability

We introduce Hoeffding inequality theorem which we later use in our analysis.

**Theorem 5** (Hoeffding Inequality). *Let $X_1, \ldots, X_m \in \mathbb{R}$ be i.i.d. random-variables with support-size bounded by $B$. Further let $\bar{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$. Then it holds for every $\varepsilon > 0$ that*

$$\Pr[|\bar{X} - \mathbb{E}[\bar{X}]| > \varepsilon] \leq 2 \cdot e^{-\frac{2m\varepsilon^2}{B^2}}.$$

## A.2 Non-interactive CCA-Commitment

**Definition 2.** *A non-interactive tagged commitment scheme consists of a pair of efficient randomised algorithms* $\mathcal{CS} = (\mathsf{Setup}, \mathsf{Commit})$ *with the following syntax.*

- $\mathsf{Setup}(1^\lambda)$*: takes as input* $1^\lambda$ *and outputs a common reference string* $\mathtt{crs}$*.*
- $\mathsf{Commit}(\mathtt{crs}, \mathtt{addr}, m; r)$*: takes as input a common reference string* $\mathtt{crs}$*, a tag/identity* $\mathtt{addr}$*, a message* $m \in \{0,1\}^*$ *and random coins* $r$ *and outputs a commitment* $com$*. We assume without loss of generality that the commitment* $com$ *contains the tag* $\mathtt{addr}$*.*

*We say that such a commitment scheme is extractable, if there exists a pair of efficient randomised algorithms* $(\mathsf{TrapdoorSetup}, \mathsf{Extract})$ *with the syntax*

- $\mathsf{TrapdoorSetup}$*: takes as input the security parameter* $1^\lambda$ *and outputs a pair* $(\mathtt{crs}, \mathsf{td})$ *of common reference string and trapdoor,*
- $\mathsf{Extract}$*: takes as input a trapdoor* $\mathsf{td}$*, and a valid commitment* $com$ *and outputs a pair* $(m, r)$ *of message an random coins*

*such that the distributions of common reference strings generated by* $\mathsf{Setup}$ *and* $\mathsf{TrapdoorSetup}$ *are computationally indistinguishable and it holds for all identities* $\mathtt{addr}$*, messages* $m$ *and random coins that*

$$\mathsf{Extract}(\mathsf{td}, com) = (m, r),$$

*where* $(\mathtt{crs}, \mathsf{td}) \leftarrow \mathsf{TrapdoorSetup}(1^\lambda)$ *and* $com \leftarrow \mathsf{Commit}(\mathtt{crs}, \mathtt{addr}, m; r)$*.*

*We say that such a scheme is hiding under* chosen commitment attacks*, if every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *has at most negligible advantage in the following experiment.*

```
$\mathsf{EXP}_{\mathcal{A}}^{\mathcal{CS},\mathsf{CCA}}(1^\lambda)$
$(\mathtt{crs}, \mathsf{td}) \leftarrow \mathsf{TrapdoorSetup}(1^\lambda)$
$(\mathtt{addr}^*, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathtt{crs})$
$b \leftarrow_\$ \{0,1\}$
$com^* \leftarrow \mathsf{Commit}(\mathtt{crs}, \mathtt{addr}^*, m_b)$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathtt{crs}, com^*)$
return $b = b^*$

$\mathcal{O}(com)$
if $\mathtt{addr} \neq \mathtt{addr}^*$ then
  return $\mathsf{Extract}(\mathsf{td}, com)$
else
  return $\perp$
```

*where the advantage of* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}_{\mathsf{CCA}}(\mathcal{A}) = \Pr\left[\mathsf{EXP}_{\mathcal{A}}^{\mathcal{CS},\mathsf{CCA}}(1^\lambda) = 1\right] - \frac{1}{2}.$$

We remark that multi-challenge security, where the adversary gets several challenge ciphertexts under different identities follows from the above notion via a routine hybrid argument.

## A.3 Time-Lock Puzzles

We follow the definitions of Bitansky et. al [BGJ+16] and we additionally require the existence of an efficiently verifiable recovery proof.

**Definition 3.** *A time-lock puzzle scheme consists of a pair of efficient randomised algorithms* $(\mathsf{PGen}, \mathsf{PSolve}, \mathsf{PVer})$ *with the syntax*

- $\mathsf{PGen}(1^\lambda, \mathbf{T}, m)$ *takes as input a security parameter* $1^\lambda$*, a hardness-parameter* $\mathbf{T}$ *and a message* $m$*, and outputs a puzzle* $tlp$*.*
- $\mathsf{PSolve}(tlp)$ *takes as input a puzzle* $tlp$ *and outputs a message* $m$ *and a proof* $\pi$*.*
- $\mathsf{PVer}(tlp, m, \pi)$ *takes as input a puzzle* $tlp$*, a message* $m$ *and a proof* $\pi$ *and outputs a bit* $b \in \{0,1\}$*.*

*We require the following properties:*

- *Completeness: For every security parameter* $\lambda$*, difficulty parameter* $\mathbf{T}$*, message* $m$ *and every puzzle* $tlp$ *in the support of* $\mathsf{PGen}(1^\lambda, \mathbf{T}, m)$ *it holds that* $\mathsf{PSolve}(tlp)$ *outputs* $m$*.*
- *Efficiency: For every message* $m$*,* $\mathsf{PGen}(1^\lambda, \mathbf{T}, m)$ *can be computed in time* $poly(\lambda, \log(\mathbf{T}))$*. Moreover, for every* $tlp$ *in the support of* $\mathsf{PGen}(1^\lambda, \mathbf{T}, m)$ *it holds that* $\mathsf{PSolve}(tlp)$ *can be computed in time* $\mathbf{T} \cdot poly(\lambda)$*.*
- *Security: We say that* $(\mathsf{PGen}, \mathsf{PSolve})$ *is secure with gap* $e < 1$*, if there exists a* $\mathbf{T}' = \mathbf{T}'(\lambda) = poly(\lambda)$*, such that for every* $\mathbf{T} = \mathbf{T}(\lambda) = poly(\lambda)$ *with* $\mathbf{T}(\lambda) > \mathbf{T}'(\lambda)$ *and every (non-uniform) PPT-adversary* $\mathcal{A}$ *of depth/parallel complexity less than* $\mathbf{T}^e$ *it holds that* $\mathcal{A}$ *has at most negligible advantage in the following experiment. Here the advantage of* $\mathcal{A}$ *is defined as*
$$\mathsf{Adv}_{\mathsf{TLP}}(\mathcal{A}) = \Pr\left[\mathsf{EXP}_{\mathcal{A}}^{\mathsf{TLP}}(1^\lambda) = 1\right] - \frac{1}{2}.$$

```
$\mathsf{EXP}_{\mathcal{A}}^{\mathsf{TLP}}(1^\lambda)$
$b \leftarrow_\$ \{0,1\}$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot,\cdot)}(1^\lambda)$
**return** $b^* = b$
```

```
$\mathcal{O}_b(m_0, m_1)$
$tlp \leftarrow \mathsf{PGen}(1^\lambda, \mathbf{T}, m_b)$
**return** $tlp$
```

- *Soundness: For every security parameter* $\lambda$*, difficulty parameter* $\mathbf{T}$*, message* $m$ *and every puzzle* $tlp$ *in the support of* $\mathsf{PGen}(1^\lambda, \mathbf{T}, m)$ *there exists a negligible function such that for all PTT adversaries* $\mathcal{A}$ *it holds that*
$$\Pr\left[\mathsf{PVer}(tlp, m, \pi) \text{ and } m \neq m' : (\pi, m') \leftarrow \mathcal{A}(tlp)\right] \leq negl(\lambda).$$

We remark that the standard definition gives the adversary only a single query to the oracle $\mathcal{O}$. The multi-query version used here follows via a standard hybrid argument and a slight loss in the gap-parameter $e$.

## A.4 Succinct Non-interactive Arguments

The notion of succinct non-interactive arguments (SNARGs) was first introduced by Micali [Mic00].

**Definition 4.** *Let* $R : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ *be an NP-witness-relation with corresponding NP-language* $\mathcal{L} := \{x : \exists w \text{ s.t. } R(x,w) = 1\}$*. A SNARG system for* $R$ *consists of three efficient randomized algorithms* $\mathsf{crsGen}$*,* $\mathsf{P}$ *and* $\mathsf{V}$ *with the following syntax.*

- $\mathsf{crsGen}(1^\lambda)$ *takes as input the security parameter* $1^\lambda$ *and outputs a common reference string* $\mathtt{crs}$*.*
- $\mathsf{P}(\mathtt{crs}, x, w)$*: Takes as input a common reference string* $\mathtt{crs}$*, a statement* $x$ *and a witness* $w$ *and outputs a proof-string* $\pi$*.*
- $\mathsf{V}(\mathtt{crs}, x, \pi)$ *Takes as input a common reference string* $\mathtt{crs}$*, a statement* $x$ *and a proof-string* $\pi$ *and outputs a verdict* $b \in \{0,1\}$*.*

*We require the following properties.*

- *Completeness: We say that* $(\mathsf{crsGen}, \mathsf{P}, \mathsf{V})$ *is complete, if it holds for all pairs* $(x, w)$ *with* $R(x, w) = 1$ *that* $\mathsf{V}(\texttt{crs}, x, \pi) = 1$ *if* $\texttt{crs} \leftarrow \mathsf{crsGen}(1^\lambda)$ *and* $\pi \leftarrow \mathsf{P}(\texttt{crs}, x, w)$*.*
- *Soundness: We say that* $(\mathsf{crsGen}, \mathsf{P}, \mathsf{V})$ *is (adaptively) sound, if it holds for every PPT-prover* $\mathsf{P}^*$ *that*
$$\Pr\left[\begin{matrix} x \notin \mathcal{L} \text{ and } \mathsf{V}(\texttt{crs}, x, \pi^*) = 1 : \\ \texttt{crs} \leftarrow \mathsf{crsGen}((1^\lambda); \pi^* \leftarrow \mathsf{P}^*(\texttt{crs}); \end{matrix}\right] \leq negl(\lambda)$$
- *Succinctness: We say that a SNARG is* succinct *if the size of the proof* $\pi$ *does not depend on the size of the statement* $x$ *nor the witness* $w$*.*

### A.5 Unlinkable Transactions

Here we formally define the concept of unlinkability for blockchain transactions.

**Definition 5.** *The* $\Gamma.\mathsf{postUnlinkTx}$ *interface is* unlinkable *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function negl such that*

$$\left| \frac{1}{2} - \Pr\left[\mathsf{UnlinkExp}^b_{\mathcal{A}}(1^\lambda) = 1\right] \right| \leq negl(\lambda)$$

*where* $\mathsf{UnlinkExp}^b_{\mathcal{A}}(1^\lambda)$ *is defined below.*

```
$\mathsf{UnlinkExp}^b_{\mathcal{A}}(1^\lambda)$
$(\mathcal{CH}, \mathsf{addr}_{R_0}, \mathsf{addr}_{R_1}, \mathsf{addr}_{S_0}, \mathsf{addr}_{S_1}, val) \leftarrow \mathcal{A}(1^\lambda)$
$\Gamma.\mathsf{postUnlinkTx}(\mathsf{payTx}, (\mathsf{addr}_{S_0}, \mathsf{addr}_{R_b}, val))$
$\Gamma.\mathsf{postUnlinkTx}(\mathsf{payTx}, (\mathsf{addr}_{S_0}, \mathsf{addr}_{R_{1-b}}, val))$
$\mathcal{CH}^* \leftarrow \Gamma.\mathsf{getChain}$
$b' \leftarrow \mathcal{A}(\mathcal{CH}^*)$
return $b = b' \wedge (\mathsf{balance}(\mathcal{CH}, \mathsf{addr}_{S_0}) \geq val)$
                $\wedge (\mathsf{balance}(\mathcal{CH}, \mathsf{addr}_{S_1}) \geq val)$
```

## B Security Analysis

Throughout the analysis we denote any function that is negligible in the security parameter by $negl(\lambda)$. We say that an algorithm is PPT if it is modelled as a probabilistic Turing machine whose running time is bounded by some function $poly(\lambda)$. The proof of Theorem 4 follows.

*Proof of Theorem 4.* If all parties are honest, the protocol implements a correct sequential sealed bid first-price auction.

Now fix an efficient adversary $\mathcal{A}$ which at each timestep $i$ controls a coalition $\mathcal{C}_i$ of nodes. We will assume that the coalition $\mathcal{C}_i$ is fixed over single timestep. Therefore, in abuse of notation we will refer to the adversarial coalition as $\mathcal{C}$ without specifying the index $i$. We will show that regardless of its strategy, the payoff which the adversary $\mathcal{A}$ achieves when participating the waiting-time auction protocol is at most a negligible amount higher than what it could achieve in a mediated auction.

As a consequence, we can conclude that honest bidding constitutes a subgame-perfect Nash-equilibrium for the waiting-time auction protocol, as honest bidding constitutes a subgame-perfect Nash-equilibrium for the mediated game. Thus, deviating from the honest strategy provides at most a negligible increase in payoff for $\mathcal{C}_i$.

Each of our changes in the following hybrid-argument only affects a single timestep. Let $\hat{L} = poly(\lambda)$ be an upper bound on the number of rounds that the system will run. Let $i'$ in the following always denote $i' = \hat{L} - i$.

**Hybrid $\mathcal{H}_0$:** This is the real experiment, i.e. the waiting-time auction implemented on a blockchain.

**Hybrid $\mathcal{H}_1$:** Identical to experiment $\mathcal{H}_0$, except that the common reference string $\texttt{crs}_{\mathsf{com}}$ is generated via $(\texttt{crs}_{\mathsf{com}}, \mathsf{td}) \leftarrow \mathsf{TrapdoorSetup}(1^\lambda)$. In other words, the common reference string of the commitment scheme is now generated together with an extraction-trapdoor $\mathsf{td}$. Clearly, hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable given the indistinguishability of modes for the commitment scheme. Therefore, the payoff of $\mathcal{C}$ in $\mathcal{H}_1$ can be at most a negligible amount higher than in $\mathcal{H}_0$.

Note the if the commitments used are modelled in the RO-model, then this step can be omitted as there is no common reference string.

For $i = 0, \ldots, \hat{L} - 1$ define the following hybrids.

**Hybrid $\mathcal{H}_{i,1}$:** In this experiment, when nodes in $\mathcal{C}$ place bids in round $i'$, their bid is extracted from the commitment $com$ using the extraction trapdoor $\mathsf{td}$. Later, when the bids are opened, the mediator disqualifies openings that are inconsistent with with the value that was extracted in the bidding-phase.

It follows from the extractability of the commitment scheme $\mathsf{Commit}$ is extractable that the payoff of $\mathcal{C}$ in $\mathcal{H}_{i-1,7}$ (or $\mathcal{H}_1$ for $i = 0$) is at most a negligible amount larger than in $\mathcal{H}_{i,1}$.

**Hybrid $\mathcal{H}_{i,2}$:** In this experiment the nodes in $\mathcal{C}$ do not submit malformed bids in round $i'$, i.e. bids for which the solving the time-lock puzzle does not provide the correct for the commitment to the bid. This holds because submitting a malformed bid does not increase the payoff of $\mathcal{C}$, but rather decrease it on average as if such a bid is detected the corresponding node loses its token of participation.

**Hybrid $\mathcal{H}_{i,3}$:** In this game the nodes in $\mathcal{C}$ are not allowed to post steal transactions for the bids of honest parties.

By the soundness property of the argument system of the time-lock puzzle, an efficient adversary has at most negligible probability in forging an accepting proof for a false statement. Therefore, the payoff of $\mathcal{C}$ does not decrease if we forbid the coalition $\mathcal{C}$ to post steal transactions.

**Hybrid $\mathcal{H}_{i,4}$:** In this game we replace the opening-phase of round $i'$ with an ideal mediated game in which the winner of the auction is determined by a mediator based in the bids extracted from the commitments $\mathsf{com}_{i'}$.

We will show in Lemma 1 that the payoff of $\mathcal{C}$ in $\mathcal{H}_{i,3}$ is larger by at most a negligible amount than in $\mathcal{H}_{i,4}$.

**Hybrid $\mathcal{H}_{i,5}$** The time-lock puzzles in the bids of honest bidders in round $i'$ are replaced with time-lock puzzles of 0. We will show in Lemma 2 that the payoff of $\mathcal{C}$ in $\mathcal{H}_{i,4}$ is larger by at most a negligible amount than in $\mathcal{H}_{i,5}$.

**Hybrid $\mathcal{H}_{i,6}$** The commitments in bids of honest nodes in round $i'$ are replaced with commitments to 0.

We will show in Lemma 3 that the payoff of $\mathcal{C}$ in $\mathcal{H}_{i,3}$ is larger by at most a negligible amount than in $\mathcal{H}_{i,4}$.

**Hybrid $\mathcal{H}_{i,7}$** Round $i'$ of the protocol is entirely handled by the mediator.

In $\mathcal{H}i, 6$ the coalition $\mathcal{C}$ obtains no information about the bids of the honest nodes via the transactions $(\texttt{addr}_j, tlp_j, com_j, \pi_j)$ and provides its bids independently of the bids of the honest parties. Therefore, the only way to increase its payoff at this point is by suppressing bids. We will show in Lemma 4 that suppressing bids does not increase the payoff of $\mathcal{C}$. Consequently, the payoff of $\mathcal{C}$ in $\mathcal{H}_{i,7}$ is not smaller than in $\mathcal{H}_{i,6}$. □

**Lemma 1.** *Given that the argument system* $(\mathsf{crsGen}_{\mathsf{win}}, \mathsf{P}_{\mathsf{win}}, \mathsf{V}_{\mathsf{win}})$ *is sound, the payoff of* $\mathcal{C}$ *in* $\mathcal{H}_{i,4}$ *is larger by at most a negligible amount than in* $\mathcal{H}_{i,3}$*.*

*Proof.* Assume towards contradiction that the payoff of $\mathcal{C}$ in $\mathcal{H}_{i,2}$ is larger by a non-negligible amount $\varepsilon$ than in $\mathcal{H}_{i,1}$. We will construct a PPT-adversary $\mathcal{A}$ that breaks the soundness of the argument system $(\mathsf{crsGen}_{\mathsf{win}}, \mathsf{P}_{\mathsf{win}}, \mathsf{V}_{\mathsf{win}})$ with non-negligible advantage $\varepsilon'$.

First note that by assumption in all rounds following round $i$ the auctions are implemented by a trusted mediator.

Conditioned on the event that the statement that announces the winner of the auction is true, deviating from the protocol in the opening-phase of round $i$ does not increase the payoff of $\mathcal{C}$. Consequently, $\mathcal{C}$ must convince the honest nodes of a false statement with non-negligible probability $\varepsilon'$. The adversary $\mathcal{A}'$ against the soundness of the argument system proceeds as follows. $\mathcal{A}'$ first gets as input a common reference string $\mathtt{crs}^*$. $\mathcal{A}$ now simulates $\mathcal{H}_{i,3}$ faithfully until the end of the opening-phase of round $i'$, except that instead of computing $\mathtt{crs}_{\mathsf{win}}$ by $\mathtt{crs}_{\mathsf{win}} \leftarrow \mathsf{crsGen}(1^\lambda)$ it sets $\mathtt{crs}_{\mathsf{win}} \leftarrow \mathtt{crs}^*$. $\mathcal{A}$ now fetches the transaction that announces the winner and outputs the statement $x = (\mathtt{crs}_{\mathsf{com}}, (com_1, \mathtt{addr}_1), \ldots, (com_\ell, \mathtt{addr}_\ell)), bid^\star, \mathtt{addr}^\star)$ and proof $\pi_{\mathsf{win}}$.

It follows immediately that with probability $\varepsilon'$ it holds that both the statement $x$ is invalid and $(x, \pi_{\mathsf{win}})$ is accepted by the verifier $\mathsf{V}_{\mathsf{win}}$. This, however, contradicts the soundness of $(\mathsf{crsGen}_{\mathsf{win}}, \mathsf{P}_{\mathsf{win}}, \mathsf{V}_{\mathsf{win}})$. □

**Lemma 2.** *Assuming that the hardness-parameter* $\mathbf{T}$ *is set such that* $\mathbf{T}^e$ *is longer than the bidding phase and given that the time-lock puzzle* $\mathsf{TLP}$ *is secure with gap* $e < 1$*, the payoff of* $\mathcal{C}$ *in* $\mathcal{H}_{i,5}$ *is at most a negligible amount smaller than in* $\mathcal{H}_{i,4}$*.*

*Proof.* Assume towards contradiction that $p(\mathcal{H}_{i,5}, \mathcal{C}) \leq p(\mathcal{H}_{i,4}, \mathcal{C}) - \varepsilon$ for a non-negligible $\varepsilon$.

First note that from the view of $\mathcal{A}$ the hybrids $\mathcal{H}_{i,4}$ and $\mathcal{H}_{i,5}$ are distinguishable after $\mathbf{T}$ has elapsed, as it can trivially open the time-lock puzzles. However, by then the bidding phase of round $i'$ is over and the auctions in the remaining rounds are implemented by the mediator.

To facilitate notation, write $p_4 = p(\mathcal{H}_{i,4}, \mathcal{C})$ and $p_5 = p(\mathcal{H}_{i,5}, \mathcal{C})$, i.e. we have $p_5 \leq p_4 - \varepsilon$. We will construct a $\mathbf{T}^e$-bounded distinguisher $\mathcal{D}$ that breaks the security of the time-lock puzzle scheme $(\mathsf{PGen}, \mathsf{PSolve})$ with advantage negligibly close to 1.

Before we provide the actual distinguisher, we will briefly sketch the idea behind this reduction. The main idea is that distinguisher $\mathcal{D}$ that plays the $\mathsf{TLP}$-distinguishing experiment can compute an *approximation* of the expected payoff of the coalition $\mathcal{C}$ by running $m$ instances of the $i$-th round of experiment $\mathcal{H}_{i,4}$ in parallel. This does not affect the parallel complexity, i.e. $\mathcal{D}$ can perform this simulation with the same parallel complexity as the $i$-th round of $\mathcal{H}_{i,4}$, which by assumption is $\mathbf{T}^e$-bounded. Moreover, computing the average of $m$ numbers can be performed in parallel complexity $poly(\log(m))$, therefore this additional step does not increase the parallel complexity by a significant amount. The distinguisher we construct will be non-uniform, specifically it will receive a transcript of the first $i' - 1$ rounds and the expectations $p_4$ and $p_5$ conditioned on this transcript as non-uniform advice.

The experiments $\mathcal{H}_{i,4}$ and $\mathcal{H}_{i,5}$ are perfectly identical until round $i'$. Therefore, if $\mathsf{tr}$ is a transcript of $\mathcal{H}_{i,4}$ for the first $i'$ rounds, we can extend $\mathsf{tr}$ to both a full transcript of $\mathcal{H}_{i,4}$ and $\mathcal{H}_{i,5}$. Let $p_{4,\mathsf{tr}}$ be the expected payoff of $\mathcal{C}$ in the $i'$-th round of $\mathcal{H}_{i,4}$ when the transcript of the first $i' - 1$ rounds is $\mathsf{tr}$. Likewise, let $p_{5,\mathsf{tr}}$ be the expected payoff of $\mathcal{C}$ in the $i'$-th round of $\mathcal{H}_{i,5}$ when the transcript of the first $i' - 1$ rounds is $\mathsf{tr}$. Clearly, it holds that $\mathbb{E}_{\mathsf{tr}}[p_{4,\mathsf{tr}}] = p_4$ and $\mathbb{E}_{\mathsf{tr}}[p_{5,\mathsf{tr}}] = p_5$. Therefore, by linearity of expectation it holds that

$$\mathbb{E}_{\mathsf{tr}}[p_{4,\mathsf{tr}} - p_{5,\mathsf{tr}}] = p_4 - p_5 \geq \varepsilon.$$

By the averaging principle, for every security parameter $\lambda$ there exists a $\mathsf{tr}^* = \mathsf{tr}^*(\lambda)$ such that

$$p_{4,\mathsf{tr}^*} - p_{5,\mathsf{tr}^*} \geq \varepsilon$$

Now fix such a transcript $\mathsf{tr}^*$. Let $R = \mathit{poly}(\lambda)$ be an upper bound for the payoff the coalition $\mathcal{C}$ can obtain in round $i$ of either $\mathcal{H}_{i,4}$ or $\mathcal{H}_{i,5}$ if the transcript of the first $i' - 1$ rounds is $\mathsf{tr}^*$. Let $m = \lceil 2\lambda \cdot \frac{R^2}{\varepsilon^2} \rceil$, which, as $\varepsilon$ is non-negligible can be upper-bounded by a polynomial for infinitely many $\lambda$. In the following we are only concerned with $\lambda$ for which $m = m(\lambda)$ is polynomially bounded.

For security parameter $\lambda$, the distinguisher $\mathcal{D}$ receives $\mathsf{tr}^*$, $p_{4,\mathsf{tr}^*}$, $p_{5,\mathsf{tr}^*}$ and $m$ as non-uniform advice. The distinguisher $\mathcal{D}$ simulates $m$ executions of the $i'$-th round of $\mathcal{H}_{i,4}$ (for starting-transcript $\mathsf{tr}^*$) in parallel, with the following modification. Each time an honest node $N_j$ wants to compute a time-lock puzzle via $\mathit{tlp}_j \leftarrow \mathsf{PGen}(\mathbf{T}, \mathit{bid}_j)$, $\mathcal{D}$ sends the message pair $(\mathit{bid}_j, 0)$ to its time-lock puzzle oracle, and sets $\mathit{tlp}_j$ to the output of the oracle. Once the $i'$-th round of all $m$ parallel simulations is completed, the distinguisher $\mathcal{D}$ computes the payoff $p^{(l)}$ of the coalition $\mathcal{C}$ for all parallel executions with indices $l \in [m]$. Next, $\mathcal{D}$ computes the empirical average $\bar{p} = \frac{1}{m}\sum_{l=1}^{m} p^{(l)}$. Finally, $\mathcal{D}$ checks if $\bar{p}$ is closer $p_{4,\mathsf{tr}^*}$ or $p_{5,\mathsf{tr}^*}$. If it is closer to $p_{4,\mathsf{tr}^*}$ it outputs 0, otherwise 1.

First notice that parallel execution-time of $\mathcal{D}$ is only marginally longer than that of the $i$-th round of $\mathcal{H}_{i,4}$. Thus, $\mathcal{D}$ is $\mathbf{T}^e$-bounded. We will now compute the advantage of $\mathcal{D}$. If the secret bit $b$ of the $\mathsf{TLP}$-experiment is 0, then each of the parallel executions simulated by $\mathcal{D}$ faithfully simulates the $i$-th round of $\mathcal{H}_{i,4}$ for transcript $\mathsf{tr}^*$. Likewise, if the bit $b$ is 1, then each parallel execution of $\mathcal{D}$ faithfully simulates the $i'$-th round of $\mathcal{H}_{i,5}$ for transcript $\mathsf{tr}^*$. We will now analyze the case $b = 0$, the case $b = 1$ follows analogously.

Therefore, assume in the following that $b = 0$. As each parallel execution faithfully simulates $\mathcal{H}_{i,4}$ for transcript $\mathsf{tr}^*$, it holds for all $l \in [m]$ that $\mathbb{E}[p^{(l)}] = p_{4,\mathsf{tr}^*}$. Consequently, by linearity of expectation we have that

$$\mathbb{E}[\bar{p}] = \frac{1}{m}\sum_{l=1}^{m} \mathbb{E}[p^{(l)}] = p_{4,\mathsf{tr}^*}.$$

As the support of each $p^{(l)}$ (for $l \in [m]$) is upper-bounded by $R$, the Hoeffding-inequality (Theorem 5) yields that

$$\Pr[|\bar{p} - p_{4,\mathsf{tr}^*}| \geq \varepsilon/2] \leq 2e^{-2\frac{m\varepsilon^2}{4 \cdot R^2}} \leq 2e^{-\lambda},$$

where the second inequality follows from $m \geq 2\lambda \cdot \frac{R^2}{\varepsilon^2}$. Consequently, except with probability $2e^{-\lambda}$, it holds that $|\bar{p} - p_{4,\mathsf{tr}^*}| < \varepsilon/2$.

Likewise, if $b = 1$ we can show that $|\bar{p} - p_{5,\mathsf{tr}^*}| < \varepsilon/2$. We conclude that the guess that $\mathcal{D}$ outputs is correct, except with probability $2e^{-\lambda}$. Thus, the advantage of $\mathcal{D}$ is

$$\mathsf{E}(\mathcal{D}) = \Pr[\mathcal{D}^{\mathcal{O}_1} = 1] - \Pr[\mathcal{D}^{\mathcal{O}_0} = 1] = 1 - 4e^{-\lambda},$$

which contradicts the security of the time-lock puzzle scheme $(\mathsf{tlpGen}, \mathsf{tlpSolve})$. □

**Lemma 3.** *From the view of $\mathcal{C}$, the experiments $\mathcal{H}_{i,5}$ and $\mathcal{H}_{i,6}$ are computationally indistinguishable by the CCA-hiding property of the commitment scheme* $(\mathsf{Setup}, \mathsf{Commit})$*. Consequently, the payoff of $\mathcal{C}$ in $\mathcal{H}_{i,6}$ is at most an negligible amount smaller than in $\mathcal{H}_{i,5}$.*

*Proof.* Assume towards contradiction that $\mathcal{A}$ distinguishes between $\mathcal{H}_{i,5}$ and $\mathcal{H}_{i,6}$ with non-negligible advantage $\varepsilon$.

The adversary $\mathcal{A}'$ against the hiding property simulates $\mathcal{H}_{i,5}$ faithfully, with the following modifications:

1. It sets $\mathtt{crs}_{\mathsf{com}} = \mathtt{crs}^*$, where $\mathtt{crs}^*$ is $\mathcal{A}$'s input.
2. Instead of extracting the commitments of nodes in $\mathcal{C}$ via the extraction trapdoor, $\mathcal{A}'$ uses the extraction oracle provided by the CCA experiment to extract these bids.
3. Instead of computing the commitments $com_j = \mathsf{Commit}(\mathtt{crs}_{\mathsf{com}}, \mathtt{addr}_j, bid_j)$ of the honest nodes by itself, it submits $\mathtt{addr}_j$ and $(bid_j, 0)$ as challenge-ciphertexts and sets $com_j$ to the challenge-ciphertext.

In the end, $\mathcal{A}'$ outputs the view of $\mathcal{A}$.

Clearly, if the secret bit $b$ of the hiding-experiment is 0, then the view of $\mathcal{A}$ in $\mathcal{A}'$'s simulation is distributed identically to $\mathcal{A}$'s view in $\mathcal{H}_{i,5}$, as the commitments $com_j$ computed by the oracle are of the form $com_j = \mathsf{Commit}(\mathtt{crs}^*, \mathtt{addr}_j, bid_j)$. On the other hand, if the secret bit $b$ is 1, then the view of $\mathcal{A}$ in $\mathcal{A}'$'s simulation is identically distributed to the view of $\mathcal{A}$ in $\mathcal{H}_{i,6}$, as the $com_j$ are of the form $com_j = \mathsf{Commit}(\mathtt{crs}^*, \mathtt{addr}_j, 0)$.

It follows that the advantage of $\mathcal{A}'$ against the hiding-experiment is identical to the advantage of $\mathcal{A}$ distinguishing between $\mathcal{H}_{i,5}$ and $\mathcal{H}_{i,6}$, which is $\varepsilon$. However, as $\varepsilon$ is non-negligible this contradicts the hiding property of $(\mathsf{Setup}, \mathsf{Commit})$. □

**Lemma 4.** *Given that the addresses used by honest bidders in round $i'$ are are fresh and unlinkable to the addresses used in prior rounds, and given that it holds for the reward $R$ and the transaction fee $F$ that $R \leq \ell \cdot F$, it is not rational for the coalition $\mathcal{C}$ to suppress bids by honest parties. Consequently, under these conditions it holds that $p(\mathcal{H}_{i,6}, \mathcal{C}) \leq p(\mathcal{H}_{i,7}, \mathcal{C})$.*

*Proof.* In this parameter-setting, we will show that the revenue lost by transaction fees is greater than the additional revenue obtained from the reward. The main idea is that since the addresses of the honest bidders are unlinkable to the addresses used in prior rounds, the adversary must blindly suppress a large number of bids to be certain that he suppresses a higher bid. However, this will reduce his revenue in transaction fees by more than what he gains by winning the newly minted coin.

Assume henceforth that there are $\ell$ honest bidders, of which $k$ bid higher than the adversarial coalition $\mathcal{C}$. Denote the (index-) set of bidders that bid higher than $\mathcal{C}$ by $V$. Assume further that $\mathcal{C}$ suppresses $t$ bids, denote the (index-) set of suppressed bids by $W$. Since the addresses used by honest bidder in this round are unlinkable to the addresses used in previous rounds, the adversary must blindly suppress $t$ out of $\ell$ bids, which in effect means that the set $W$ is chosen uniformly at random. There are $\binom{\ell}{t}$ possible choices for the set $W$. Out of these, $\binom{\ell-k}{t-k}$ contain the (fixed) set $V$. Thus it holds that

$$\Pr_W[V \subseteq W] = \frac{\binom{\ell-k}{t-k}}{\binom{\ell}{t}} = \frac{(\ell-k)! \cdot t!}{\ell! \cdot (t-k)!}. \tag{1}$$

If the adversary suppresses a set of bids $W$ that contains $V$, then it will win the reward $R$, but lose an amount of $t \cdot F$ in transaction fees. On the other hand, if $W$ does not contain $V$, then the change in payoff for the adversary only consists in the loss of $t \cdot F$ in transaction fees. Therefore, in expectation the difference in the adversary's payoff compared to following the honest strategy is

$$\Delta\tilde{p}_t = \Pr_W[V \subseteq W] \cdot R - t \cdot F = \frac{(\ell-k)! \cdot t!}{\ell! \cdot (t-k)!} \cdot R - t \cdot F.$$

We will now show that under the condition $R \leq t \cdot F$ the value $\Delta\tilde{p}_t$ is always $\leq 0$, which means that suppressing bids does not increase the payoff for the adversary. It holds that

$$\begin{aligned}
\Delta\tilde{p}_t &\leq \frac{(\ell-k)!\cdot t!}{\ell!\cdot(t-k)!}\cdot\ell\cdot F - t\cdot F \\
&= t\cdot F\cdot\left(\frac{(\ell-k)!\cdot(t-1)!}{(\ell-1)!\cdot(t-k)!}-1\right) \\
&= t\cdot F\cdot\left(\underbrace{\frac{((\ell-1)-(k-1))!\cdot(t-1)!}{(\ell-1)!\cdot((t-1)-(k-1))!}}_{=:\gamma}-1\right).
\end{aligned}$$

Recall that $k > 0$. By equation 1 the term $\gamma$ is the probability that a random subset of size $t-1$ of a set of size $\ell-1$ contains a fixed set of size $k-1$. As $\gamma$ is the probability of an event it holds that $\gamma \leq 1$ and consequently $\Delta\tilde{p}_t \leq 0$.

Consequently, if one of these two points holds, suppressing bids in round $i'$ of $\mathcal{H}_{i,6}$ does not increase $\mathcal{C}$'s payoff, and we can conclude that $p(\mathcal{H}_{i,6},\mathcal{C}) \leq p(\mathcal{H}_{i,7},\mathcal{C})$. This concludes the proof. $\square$