Virtual ASICs: Generalized Proof-of-Stake Mining in Cryptocurrencies

Chaya Ganesh[1]*, Claudio Orlandi[2]**, Daniel Tschudi[3]*, and Aviv Zohar[4]***

chaya@iisc.ac.in,orlandi@cs.au.dk,dt@concordium.com,avivz@cs.huji.ac.il

[1] Indian Institute of Science
[2] Concordium Blockchain Research Center, Aarhus University, Denmark
[3] Concordium, Zurich
[4] The Hebrew University of Jerusalem

Abstract. In proof-of-work based cryptocurrencies, miners invest computing power to maintain a distributed ledger. The drawback of such a consensus protocol is its immense energy consumption. Bitcoin, for example consumes as much energy as a small nation state. To prevent this waste of energy various consensus mechanism such as proof-of-space or proof-of-stake have been proposed. In proof-of-stake, block creators are selected based on the amounts of money they stake instead of their expanded computing power.
In this work we study *Virtual ASICs*–a generalization of proof-of-stake. Virtual ASICs are essentially a virtualized version of proof-of-work. Miners can buy on-chain virtual mining machines which can be powered by virtual electricity. Similar to their physical counterparts, each powered virtual ASIC has a certain chance to win the right to create the next block. In the boundary case where virtual electricity is free, the protocol corresponds to proof-of-stake using an ASIC token which is separate from the currency itself (the amount of stake equals your virtual computing power). In the other boundary case where virtual computers are free, we get a proof-of-burn equivalent. That is, a consensus mechanism in which miners 'burn' money to obtain lottery tickets for the right to create the next block. We provide the cryptographic machinery required to base a consensus protocol on Virtual ASICs, as well as to sell them in sealed-bid auctions on-chain. We ensure that as long as a majority of the miners in the system mine honestly, bids remain both private and binding, and that miners cannot censor the bids of their competitors. To achieve this, we introduce a novel all-or-nothing broadcast functionality in blockchains that is of independent interest.

1 Introduction

Nakamoto's blockchain protocol [Nak08] crucially relies on a *proof-of-work (PoW)* component to secure the chain. In order to make the system secure against double spending attacks, honest miners are encouraged to join the system via high valued block rewards. The competitive nature of mining quickly led miners to use specialized hardware for the mining process. Proof-of-work based mining nowadays uses Application Specific Integrated Circuits (ASIC) to perform the required computations. Since manufacturing and operating ASICs requires resources such as electricity to power and cool the ASICs, Bitcoin's mining exacts a high toll on the world's resources [OM14, CDE$^+$16]. Since the economics of mining suggests that miners should purchase more machines as long as rewards outweigh costs, more efficient mining equipment still wastes just as many resources at equilibrium [TSE19]. For the same reason, other variants of proof-of-work, e.g., those that require more memory such as Ethash [Eth] or even processor down time or ownership [MHWK16], lead to waste at a similar large scale.

* Work partially done when authors were at Aarhus University supported by the Concordium Blockhain Research Center.

** Claudio Orlandi is supported by: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC).

*** Aviv Zohar is supported by the Israel Science Foundation (grant 1504/17) and by a grant from the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau.

The wasteful nature of PoW led to the exploration of alternative consensus mechanisms. *Proof-of-stake consensus protocols (PoS)* provides guarantees similar to PoW assuming that a majority of the *wealth* in the system is controlled by honest participants (as opposed to a majority of the *computing power* being honest). The rationale is that users who have a significant *stake* in the system have an incentive to keep the system running. Otherwise, they risk a devaluation of their holdings.

Starting from the initial idea of proof-of-stake in an online Bitcoin forum [bit11], there have been a series of candidates for such PoS protocols [KN12, BLMR14, BGM16]. More recently, there have been works on formal models for proof-of-stake and constructions with provable security guarantees [BPS16, KRDO17, GHM$^+$17, DGKR18, BGK$^+$18].

In this paper we argue that the design space of PoS-like protocols can be expanded and that further improvements are possible. We start by discussing some advantages and disadvantages of PoW and PoS and show how the best of both worlds can be combined with a system based on *virtual ASICs*. Virtual ASICs are tokenized representations of mining power that mimic many of the properties of their physical counterparts but do not waste any physical resources. Their properties can be fine-tuned to further adjust the incentives of miners in ways that improve the stability, well-being, and decentralization of the system.

1.1 Our Contributions

We present the following technical contributions:

Consensus based on virtual ASICs. We construct a consensus protocol that is based on a leader election lottery, where the probability that a party is elected as a leader in a given slot is proportional to the party's mining rate of the virtual ASICs in the system. The idea of replacing stake with virtual ASIC power in a lottery will generalize to other proof-of-stake protocols.

On-chain auctions for ASICs. We show how to bootstrap our ASIC blockchain system by constructing a mechanism for acquiring ASICs on the blockchain. Towards this, we construct a sealed-bid auction that, in addition to bid confidentiality, independence and verifiabilty, provides *censorship resilience*, that is, parties cannot be prevented from bidding on ASICs even by other miners, and unretractability, that is, the winner of the auction cannot retract their bid in the decision stage of the auction. In constructing our auction protocol, we first construct a building block that we call an *all-or-nothing broadcast.*

All-or-nothing broadcast channel. At a high level, an all-or-nothing broadcast channel allows parties to broadcast a message with delay. In the first phase, parties can input messages, but no party can retrieve them. After the delay time has elapsed, parties can retrieve all the messages in the second phase. The guarantee is that an adversary can (at most) prevent the delivery of all messages, but the delivery of any specific message cannot be prevented. We use this channel to construct an ASIC auction, and also to allow miners to commit in advance to powering their ASICs. We believe all-or-nothing broadcast on the blockchain could be of independent interest for other applications. We use an ad hoc threshold encryption scheme to construct our channel, and then discuss how to use rewards to disincentivize aborts, thus achieving a delay broadcast which guarantees output delivery.

1.2 The Advantages and Disadvantages of Physical ASICs vs. PoS

In addition to their infamously high resource consumption, ASICs introduce additional problems compared to PoS systems. Specifically, they have side-effects that lead to more centralization in the system, and to miner behavior that causes mining power to oscillate between different cryptocurrencies based on profitability.

ASICs as a centralizing force. ASIC based PoW systems, are particularly efficient in mining relative to general computing platforms such as CPUs, GPUs or even FPGAs. When they first appeared, ASICs were seen as a potential threat to the decentralization of Bitcoin: They conferred substantial advantages to the few miners that possessed them. These miners could then easily push others below the threshold of profitability, and essentially control the cryptocurrency. Similar concerns were raised with respect to the manufacturing of ASICs that had a high barrier of entry and is thus controlled by even fewer parties.

Another such barrier is represented by patents on Hardware designs and methods that allow for even more efficient mining. One such example is ASIC Boost, a technique for running SHA256 hashes more efficiently in hardware. When ASIC Boost appeared, it was feared that patent protections would be used to restrict the profitability of competing miners and again allow only a single entity to mine profitably.

ASIC-resistant mining schemes were explored in attempt to alleviate these issues, but were also met with objections. Arguably, ASIC resistance raises the barrier to create an efficient mining machine, which implies that once that barrier is breached, it will be in even fewer hands. For example, Ethereum purposefully chose a PoW system based on intensive memory access [Eth], claiming that memory available on the market is already extremely efficient, and it is unlikely that some single ASIC manufacturer would deploy an ASIC that is substantially better that other miners. Nonetheless, ASICs for Ethereum did eventually appear (although their mining efficiency is not as far from conventional miners as was the case in Bitcoin).

Finally, the operation of ASICs often leads to centralization via an effect of "Economies of Scale". Large miners are able to obtain ASICs, electricity, and other resources at far lower prices. Their large size often confers a discount on operating costs. This again translates to higher profits than small miner.

Needless to say, PoS systems as well as our Virtual ASICs do not suffer from any of these disadvantages; they work just as efficiently for small miners, and are available to all at the same price.

The advantage of ASICs. One often cited advantage of ASICs (Application Specific Integrated Circuits) is the very fact that they are, as the Acronym ASIC suggests, application specific. The owners of ASICs are in this way invested heavily in the cryptocurrency—their ASICs are useless for any other purpose. This is a major advantage for cryptocurrency security because any miner that holds a large amount of ASICs would refrain from using them to attack the protocol, due to the fear of lost future earnings that the ASICs would have otherwise provided, coupled with the inability to sell or re-purpose the ASICs.

Other general mining equipment such as GPUs have in the past been repurposed and resold when, e.g., the exchange rate of Ethereum plumetted (GPUs were sold back into the second hand parts market mainly to gamers causing record losses for GPU manufacturers on the following year).[5] We note that the existence of other cryptocurrencies that utilize the same PoW mechanisms offers some possibility to repurpose mining equipment (to mine on these alternative cryptocurrencies). Such is the case with many of the Bitcoin forks such as "Bitcoin Cash".

Since our Virtual ASICs are held "on chain", they cannot be used to mine other cryptocurrencies. They thus represent an idealized form of ASIC that is truly useless outside the confines of the system. They therefore achieve the same effect, tying the future rewards of miners to the success of the system.

Borrowing money and rental attacks—the weakness of PoS. Proof-of-Stake systems seem to offer the right incentives: those holding many tokens are also those that have the most to lose: if the system is crippled by an attack, one could argue that the value of their tokens will be lost. While this seems on the surface to align incentives of miners with the good of the system, it is not necessarily the case. The primary reason is that in any well functioning economic system, it is possible to borrow money. Thus, an attacker that borrows money for the period of one year, could use these funds as stake and attack the system. At the end of the year, the stake can be returned, and thus the attacker is left unaffected by any loss of value of the tokens (this attacker is essentially also shorting the currency he is attacking, which offsets any loss due to devaluation).

A similar observation was made in the ASIC market: if rental of large amounts of ASICs is possible, the cost to attackers is just the cost of renting equipment [Bon16]. Arguably, this is the reason why miners would never loan out large amounts of equipment for potential use by attackers. While mining contracts can be drawn up to provide someone slices of the mining profits, the owner of miners typically retains control of their operation and thus ensures they are not used for attacks.

Our proposal for Virtual ASICs separates the mining tokens from the currency. We thus achieve an effect similar to that of ASICs: transferring a Virtual ASIC to another party allows it to attack the system, and therefore miners ought to refuse to loan out mining equipment (selling it for its full market price is okay). A market for loans in the currency token of the system can still exist, including the ability to short the currency.

[5] `https://www.overclock3d.net/news/gpu_displays/used_gpus_flood_the_market_as_ethereum_s_price_crashes_below_150/1`

1.3 Related Work

Time-elapsed cryptography. Our notion of the all-or-nothing broadcast channel is reminiscent of Time-lapse cryptography [RT06]. In [RT06] too, the goal is to send a message to the future so that it can be decrypted after a specified amount of time has elapsed. However, the service that is built in the work of [RT06] requires a distributed key generation process among the participants. In our setting, it is prohibitive to have to know the set of participants in advance. In addition, we would like the key generation to be a local ad hoc process that does not involve communication. In our construction, parties just publish one message as a public key component and at release time, publish a private key. Our construction also exploits the ledger property of the blockchain in order to achieve broadcast with all or nothing property such that all the messages sent to the channel are revealed after a specific amount of time has elapsed, or none of them are.

Time-lock puzzles introduced in [RSW96] allow one to encrypt messages for the future, by generating a puzzle together with a solution that remains hidden until the specified time has elapsed. However, the the hardness parameter needs to be set conservatively large so as to avoid the secrets becoming public sooner than desired. Fine-tuning the parameterization of a time-lock puzzle in practice requires estimating adversarial time. Thus, the parameter of the puzzle must, by design, be large (to accommodate for an adversary willing to devote computational power) thus making the release phase inefficient, since in the worst case honest parties have to compute the puzzle of all corrupt parties. The work of [MT19] construct homomorphic time-lock puzzles with application to sealed-bid auction on the blockchain. Each bidder time-locks their bid and the highest bidder is computed homomorphically over the puzzles. However, the resulting protocol requires fully-homomorphic time-lock puzzles and are not efficient in practice.

A construction with similar guarantees as our all-or-nothing broadcast in the synchronous setting is given in [HZ10]. In [HZ10], the authors construct a broadcast in the adaptive threshold adversary setting that utilizes verifiable secret sharing techniques, and can be seen as an analogue of our channel in the synchronous setting.

On-chain Auctions. In HAWK[KMS$^+$16], smart contracts are employed to run auctions on top of a blockchain. However, the existence of a trusted party is assumed to run the auction contract. This manager of the bidding contract is trusted with the bidders' inputs. In our setting, we do not have a trusted party since we need the auction to sell ASICs used in the consensus protocol of the blockchain. In [DDM$^+$18], the authors propose a protocol to run an auction over a blockchain in the context of constructing a minting mechanism based on waiting-time first-price auctions. Our use of time-lock puzzles to disincentivize aborts in the all-or-nothing broadcast protocol is similar to the use of time-lock puzzles in [DDM$^+$18] to deter parties from refusing to open their bids.

2 Preliminaries

Notation. We use $[1, n]$ to represent the set of numbers $\{1, 2, \ldots, n\}$. If A is a randomized algorithm, we use $y \leftarrow \mathsf{A}(x)$ to denote that y is the output of A on x. We write $x \overset{R}{\leftarrow} \mathcal{X}$ to mean sampling a value x uniformly from the set $\mathcal{X}$. We write PPT to denote a probabilistic polynomial-time algorithm. Throughout the paper, we use κ to denote the security parameter or level. A function is negligible if for all large enough values of the input, it is smaller than the inverse of any polynomial. We use negl to denote a negligible function. We denote by H a cryptographic hash function.

2.1 Verifiable Random Function

In our protocols, parties use a *verifiable random function* (VRF) scheme to get the randomness used in the consensus protocol.

Definition 1. *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^k \rightarrow \{0,1\}^{\ell(k)}$ is a family of VRFs, if there is a tuple of algorithms ($\mathsf{VRFkeygen}$, $\mathsf{VRFeval}$, $\mathsf{VRFverify}$) such that: $\mathsf{VRFkeygen}(1^\kappa)$ generates a key pair $(\mathsf{pk}, \mathsf{k})$; $\mathsf{VRFeval}_\mathsf{k}(\mathsf{pk}, x)$ outputs a tuple $(F_\mathsf{k}(x), \pi_\mathsf{k}(x))$ where $\pi_\mathsf{k}(x)$ is the proof of correct evaluation;* $\mathsf{VRFverify}_{\mathsf{pk}}(x, y, \pi)$ *verifies that $y = F_\mathsf{k}(x)$ using the proof π. We require that the following properties are satisfied.*

Pseudorandomness. *For any pair of PPT* $(\mathcal{A}_1, \mathcal{A}_2)$*, the following probability is* $1/2 + \mathsf{negl}(\kappa)$*.*

$$\Pr\left(\begin{array}{c|c} b = b' & (\mathsf{pk}, \mathsf{k}) \leftarrow \mathit{VRFkeygen}(1^k); \\ \wedge x \notin Q_1 \cup Q_2 & (Q_1, x, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathit{VRFeval}(\cdot)}(\mathsf{pk}); \\ & y_0 = F_{\mathsf{k}}(x); y_1 \leftarrow \{0,1\}^{\ell}; \\ & b \leftarrow \{0,1\}; (Q_2, b') \leftarrow \mathcal{A}_2^{\mathit{VRFeval}(\cdot)}(y_b, \mathsf{state}) \end{array}\right).$$

The sets Q_1, Q_2 *contain all the queries made to the evaluation oracle. The random variable* state *stores information that* $\mathcal{A}_1$ *can save and pass on to* $\mathcal{A}_2$*.*

Uniqueness. *There does not exist values* $(\mathsf{pk}, x, y_1, y_2, \pi_1, \pi_2)$ *such that:*

$$y_1 \neq y_2 \text{ and } \mathsf{VRFverify}_{\mathsf{pk}}(x, y_1, \pi_1) = \mathsf{VRFverify}_{\mathsf{pk}}(x, y_2, \pi_2) = 1$$

.

Public Verifiability: *If* $(y, \pi) = \mathit{VRFeval}_{\mathsf{k}}(\mathsf{pk}, x)$*, then* $\mathit{VRFverify}_{\mathsf{pk}'}(x, y, \pi) = 1$

Unpredictability under malicious key generation *For any pair of PPT* $(\mathcal{A}_1, \mathcal{A}_2)$*, the following probability is* $1/2 + \mathsf{negl}(\kappa)$*.*

$$\Pr\left(\begin{array}{c|c} & (\mathsf{pk}, \mathsf{k}) \leftarrow \mathcal{A}_1(1^{\kappa}); \\ b = b' & x \leftarrow \{0,1\}^n; \\ & y_0 = F_{\mathsf{k}}(x); y_1 \leftarrow \{0,1\}^{\ell}; \\ & b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}_2(y_b.\mathsf{pk}, \mathsf{k}) \end{array}\right).$$

A UC definition capturing the above requirements of a VRF scheme was given in [DGKR18], together with an instantiation under the CDH assumption in the random oracle model.

2.2 Ad hoc Threshold Encryption

We now define a threshold encryption scheme that we will use as a building block in constructing our all-or-nothing broadcast protocol. In a threshold encryption scheme, a key generation algorithm outputs a public key and a set of secret keys which are generated with respect to a threshold value t. Given any set of at least $t + 1$ secret keys, one can decrypt a ciphertext, while security guarantees that the encryption remains secure even given any set of t secret keys. Since we cannot require a trusted dealer running key generation, we use an *ad hoc threshold-encryption scheme.* In an ad hoc threshold-encryption scheme, each user has a public key and a corresponding secret key such that all the key pairs are independently generated. A sender can choose a set of recipients and a threshold on the fly and encrypt to that set using the public keys of the parties in the set such that any $(t + 1)$ parties in the set can decrypt. In the definition below, we consider the case where the recipient set is the set $\mathcal{U}$ of all participants.

Definition 2. *A* (t, n) ad hoc threshold encryption *(ATE) scheme is a tuple of algorithms* $(\mathit{KeyGen}, \mathit{Enc}, \mathit{Dec})$ *defined as follows.*

Key Generation *KeyGen is a randomized key-generation algorithm that takes the security parameter and returns a public-key private-key pair.*

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathit{KeyGen}_{\mathsf{t},\mathsf{n}}(\kappa)$$

Encryption *Enc is a randomized encryption algorithm that takes a message* m *and encrypts to the set of public-keys* $\{\mathsf{pk}_i\}_{i=1}^{\mathsf{n}}$ *in* $\mathcal{U}$ *and generates a ciphertext* c *such that any size-*$(\mathsf{t}+1)$ *subset of should be able to jointly decrypt.*

$$c \leftarrow \mathit{Enc}_{\mathsf{t},\mathsf{n}}(\mathcal{U}, m)$$

Decryption *Dec is a deterministic decryption algorithm takes a ciphertext* c*, a set* I *of size* $\mathsf{t}+1$*, the secret keys corresponding to the subset* I *and outputs the message* m*.*

$$m \leftarrow \mathit{Dec}_{\mathsf{t},\mathsf{n}}(I, \{\mathsf{sk}_j\}_{j \in I}, c)$$

A secure ATE scheme satisfies the following properties.

Correctness: *For $i \in \{1, \ldots, \mathfrak{n}\}$ let $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}_{\mathfrak{t},\mathfrak{n}}$. Let m be an arbitrary message and $c \leftarrow \mathsf{Enc}_{\mathfrak{t},\mathfrak{n}}(\mathcal{U}, m)$. Then for any $I \subset \{1, \ldots, \mathfrak{n}\}$ of size $\mathfrak{t}+1$, it holds that $m \leftarrow \mathsf{Dec}_{\mathfrak{t},\mathfrak{n}}(I, \{\mathsf{sk}_j\}_{j \in I}, c)$.*

CCA2 Security: *The scheme satisfies threshold CCA2 security defined as: for all PPT algorithms $\mathcal{A}$, the advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathfrak{n},\mathfrak{t}}$ is negligible in κ, where $\mathsf{AdvCCA}_{\mathcal{A},\mathfrak{n},\mathfrak{t}} = |\Pr(b = b^*) - \frac{1}{2}|$ denotes the advantage of adversary $\mathcal{A}$ in the game defined below.*

- *For $i \in \{1, \ldots, \mathfrak{n}\}$ let $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}_{\mathfrak{t},\mathfrak{n}}$. $\mathcal{A}$ is given $\{\mathsf{pk}_i\}_{i \in \mathcal{U}}$.*
- *$\mathcal{A}$ chooses $S \subset \mathcal{U}, |S| \leq t - 1$, and obtains sk_i for all $i \in S$.*
- *$\mathcal{A}$ can query the decryption oracle with a tuple (c, I), for a ciphertext c, a set $I \subset \{1, \ldots, \mathfrak{n}\}, |I| = \mathfrak{t}+1$ to obtain $\mathsf{Dec}_{\mathfrak{t},\mathfrak{n}}(I, \{\mathsf{sk}_j\}_{j \in I}, c)$.*
- *$\mathcal{A}$ outputs m_0, m_1 of equal length.*
- *Choose random $b \leftarrow \{0, 1\}$, $\mathcal{A}$ is given $c^* = \mathsf{Enc}_{\mathfrak{t},\mathfrak{n}}(\mathcal{U}, m_b)$.*
- *$\mathcal{A}$ chooses more decryption queries (c, I), for $c \neq c^*$, $I \subset \{1, \ldots, \mathfrak{n}\}, |I| = \mathfrak{t}+1$ and obtains $\mathsf{Dec}_{\mathfrak{t},\mathfrak{n}}(I, \{\mathsf{sk}_j\}_{j \in I}, c)$.*
- *$\mathcal{A}$ outputs b^*.*

When in the above game, the adversary has access to the decryption oracle only before seeing the challenge ciphertext, the scheme is said to satisfy CCA1 security. A straightforward way to construct such an ATE scheme is the following: KeyGen generates the key pair of an underlying public key encryption scheme. To encrypt, the encryption algorithm secret shares the message m using a $\mathfrak{t}$-out-of-n Shamir secret sharing scheme and encrypts the shares. The ciphertext is therefore an n-tuple, where the i-th component is an encryption of the i-th share under the i-th public key. In [DHMR08] and [DHMR07], ATE schemes with ciphertext size linear in $n - t$ are proposed satisfying CCA1 and CCA2 security respectively. In [RSY18], the authors show how to achieve ciphertext size that is independent of n, assuming indistinguishability obfuscation. Recent results show that under standard assumptions, it is impossible to achieve a constant size ciphertext for an ATE scheme (making black-box use of the underlying PKE scheme, and no other crypto primitive) [DY20]. In our setting, we require non-malleability, and hence the scheme of [DHMR07] is a suitable candidate.

3 Model

In this section, we define the model and functionalities that are used in the protocol section.

3.1 Time and Network

In our setting, we consider time as divided into discrete *slots*. An *epoch* consists of k slots. Parties have clocks that indicate the current slot.

We assume a network with bounded delay, which is parameterized by an upper bound Δ on the network delivery time. It allows parties to multicast messages. That is, any message sent by an honest party in round r is guaranteed to arrive at all honest parties until round $r + \Delta$. The actual delay of messages (per message and party) can be set by the adversary (within Δ). The delay Δ is *not* known to the honest parties.

3.2 Virtual ASICs

In our consensus protocol parties use virtual ASICs to win the right to create new blocks. The probability of being selected as block creator is proportional to the ASIC's mining rate of the total rate of all powered ASICs.

Definition 3. *A* virtual ASIC *is a tuple $(\mathsf{id}, \mathsf{vk}, \mathsf{pk}, \mathsf{initMR}, j_0)$ where id is the identifier, vk is the signature public key of the owner, pk is the ASIC's VRF public key, $\mathsf{initMR} \in [0, 1]$ is the initial mining rate, and j_0 is the spawn epoch, i.e., the first epoch in which the ASIC can be used.*

We use the notation $\mathsf{ASIC}.x$ to refer to property x of ASIC.

In an epoch, a virtual ASIC is either *powered* or *unpowered*. To power an ASIC in epoch j, the owner buys power on the blockchain before the previous epoch. The function $\mathsf{ispowered}$ (parametrized by the blockchain) takes as input an ASIC, and an epoch j and returns true if the ASIC is powered in epoch j. We will simply write, $\mathsf{ASIC.ispowered}_j$ to denote this output.

ASICs degrade over time. The *degradation* is discrete and occurs at the end of each epoch. The degradation rate in epoch j may depend on whether the ASIC is powered in that epoch or not. The degration is formally captured by the degradation function g which takes as input the mining rate for epoch j, and a bit indicating whether the ASIC was powered in epoch j or not. The output is the mining rate for epoch $j+1$. We denote by $\mathsf{ASIC.MR}_j$ the mining rate of ASIC in epoch j. The *actual mining rate* of an ASIC in epoch i is its mining rate if powered or 0 otherwise, i.e., $\mathsf{MR}_j \cdot \mathsf{ispowered}_j$. All the above information is assumed to be extractable from the blockchain. In particular, the actual mining rate of an ASIC is public.

Buying ASICs. In the most simple case, parties could buy new ASICs (e.g. with a fixed initial mining) by sending money to a specific payment address. It is important that parties are not censored from buying ASICs. We describe an auction protocol for ASIC in Section 5. that is resilient to censorship and allows for more complex economic models (c.f Section 6).

Buying power for ASICs. Parties need to buy power in order to use their ASIC to create blocks in an epoch. The price of power will depend on the economic model used. See Section 6 for more details on power pricing. For the consensus protocol described in Section 4.1 parties must by power for epoch j before the start of epoch $j-1$. Otherwise, parties could buy power after learning the actual winning chances of their ASICs in epoch j.

The technical solution to buying power is simple. To buy power a party sends the necessary amount to a special "power" address. The transaction contains the ID of the ASIC and the epoch in which the ASIC shall be powered. Power transactions can be subject to censorship as well, and to prevent this, one could use a similar solution as for auctions as discussed in Section 5.2. The idea is that parties first pay the money to the power address using shielded transactions and then use the all-or-nothing broadcast to announce which ASIC they want to power. The money sent to the payment and/or power address can be either burned or used later for block rewards.

A potentially simpler solution to prevent censorship is to reveal the shielded transaction when a new block is created. However, with this solution the total mining power is not known at the beginning of the epoch. Thus the lottery difficulty must be set using estimates on the total mining power from previous epoch. Note that in our consensus protocol in Section 4.1 we assume that the total mining power is known.

4 Consensus based on Virtual ASICs

At a high level, a lottery-based consensus works by implementing a publicly verifiable "lottery" mechanism that elects a committee or a leader who is allowed to produce the next block of the blockchain. The probability that a party wins the lottery is proportional to the amount of some "scarce resource" that the party owns. This scarce resource is computing power in the case of proof-of-work, and the amount of coins the party owns in the system in the case of proof-of-stake. Roughly, the scarce resource cannot be replicated to increase the probability of winning by mounting a Sybil attack. Together with the assumption that the majority of this resource is in the hands of honest parties, one gets the guarantees necessary to build consensus.

4.1 Nakamoto Style Consensus Protocol

In this section present a consensus protocol using virtual ASICs based on the template of Ouroboros Praos (OP) [DGKR18] a proof-of-stake based protocol. Roughly, in Ouroboros Praos, a leader is elected in every slot using a VRF-based leader election scheme. The probability that a party pid is elected as leader in a given slot is proportional to the party's relative stake in the system. We take this leader election scheme, but replace stake by the mining rate of the virtual ASICs. That means a party wins the leader election lottery if one of its ASICs wins. The winning probability is therefore proportional to the actual mining rate of all owned ASICs.

Initial set of ASIC and mining rate. Similar to the initial stake distribution in Ouroboros Praos, we have a set of initial virtual ASICs. We assume that these ASICs are all powered in the first two epochs.

The mining rate of any ASIC for a given epoch j is public knowledge as it can be read from the blockchain.

Slot lottery. We take the VRF-based OP slot lottery and replace stake as scarce resource by ASIC mining rate. In each epoch j the total mining rate is defined as

$$\mathsf{totalMR}_j = \sum_{\mathsf{ASIC}} \mathsf{ASIC.ispowered}_j \cdot \mathsf{ASIC.MR}_j.$$

An ASIC ASIC wins lottery for slot sl of epoch j if

$$y < 2^{\ell} \cdot \left(1 - (1-f)^{\frac{\mathsf{MR}_j}{\mathsf{totalMR}_j}}\right)$$

where $(y, \pi) = \mathsf{VRF}(\mathsf{ASIC.sk}, \mathsf{sl}||\mathsf{nonce}_j)$, $\mathsf{MR}_j = \mathsf{ASIC.ispowered}_j \cdot \mathsf{ASIC.MR}_j$, and f is the difficulty of the lottery. As in OP we use an epoch nonce nonce_j to make the lottery unpredictable for the far future. The nonces are generated from randomness in blocks of the previous epoch (see Section 5.1 in [DGKR18] for more details).

A party can locally check if one if its ASICs won using the VRF evaluation keys. If the party won with ASIC ASIC in sl it will include $(\mathsf{ASIC.id}, y, \pi)$ in the newly created block. This allows anyone to verify the winning condition.

Blocks and blocktree. We use the same *block structure* as Ouroboros Praos. A block is given by a tuple that indicates the slot, the data, a pointer to the previous block, a block proof and a signature of the entire block content under the signing key of ASIC that won the slot lottery. The block proof $(\mathsf{ASIC.id}, y, \pi)$ consists of the ASIC ID, and the necessary values to check the VRF evaluation. Roughly, a block is valid if it has the right format, contains valid data, and the creator of the block actually won the slot lottery. We can therefore use the block validity predicate from Ouroboros Praos and replace the slot lottery winning condition by the one described in the paragraph above.

Every party locally stores a *blocktree*. Initially, the blocktree consists of the genesis block. During the consensus protocol the party will extend the blocktree by all valid blocks it receives.

In the consensus protocol honest parties will try to extend their *longest chain*. Roughly, the length of a chain is a weighted sum of its blocks where the weight of a block is the difficulty of its slot lottery. Therefore, for constant difficulty the length of a chain is exactly the number of blocks. For more details see Section 5.1 in [DGKR18].

Protocol. As said above, our consensus protocol is the Ouroboros Praos protocol where we replace the stake lottery with the ASIC lottery. All the necessary changes are outlined in the above. For completeness, we give an informal description from the view of party P.

Protocol Consensus Protocol

The party P stores a blocktree $\mathcal{T}$. For any epoch j P knows the set of relevant ASICs with all their properties (including their actual mining rate.) This information can be computed directly from the longest chain in the blocktree.
In slot sl of epoch j P does the following:

1. Add all new valid blocks to the block tree $\mathcal{T}$.
2. For all owned ASICs check if they won the lottery for sl.
3. If an ASIC won the lottery create a new block that extends the longest chain in $\mathcal{T}$.

Security analysis. We can derive our security statement directly from the ones of Ouroboros Praos. In particular, for a single epoch our protocol corresponds to Ouroboros Praos where ASICs play the role of bakers and mining rate the role of stake. Thus, as long as a majority of the total mining rate is controlled by honest parties our protocol will be secure.

Lemma 1. *If a majority of* $\mathsf{totalMR}$ *is honest, the above protocol achieves common-prefix, chain-growth, and chain-quality as defined in [DGKR18].*

4.2 Consensus from Generic Proof-of-stake Protocols

Even though the consensus protocol we presented is along the lines of Ouroboros Praos, we expect that our ideas will work with other proof of stake protocols where we replace stake with virtual ASICs. This will result in a lottery based on virtual ASIC power, where the rest of the protocol remains unchanged.

5 Auction Protocol for Virtual ASICs

In a system using consensus based on virtual ASICs, we need to allow new miners a chance to join the system. We therefore need new ASICs in the system. We propose conducting auctions on-chain for such newly created virtual ASICs. Consider the following, simplistic way of implementing ASICs auctions: all parties commit to their bids on the blockchain and then, after some agreed deadline, they open the bids. This is not a satisfactory auction since it allows bids to be selectively revealed i.e., an adversary might wait for some of the other bids to be opened before choosing whether to open its own bids or not. Moreover, as the blockchain is an imperfect bulletin board, a malicious miner could prevent the bids of an honest party to appear on the blockchain, effectively *censoring* the bids of the honest parties. Therefore, we need to design an auction mechanism which is resilient to censorship. In this way honest parties will not be prevented from acquiring, and therefore owning, ASICs. We achieve this by designing a novel mechanism that might be of independent interest, which achieves an *all-or-nothing property* meaning that either all bids will be revealed or none will. Thanks to this tool we can effectively design an *all-or-nothing broadcast channel*, which in turn can be used to construct an auction protocol.

Two-stage sealed-bid auction. A sealed-bid auction, consists of a bidding phase and a decision phase, and the bids are kept secret during the bidding phase. In the bidding phase, parties can submit their bids to the auction. In the decision phase, the winner(s) and the winning price is determined. We consider a $(k+1)$-st price auction used to sell k ASICs, where the $(k+1)$-st highest bid is the winning price, and parties who bid higher than the winning price are declared winners. A $(k+1)$-st price is equivalent to the Vickrey auction for $k = 1$. In our application, we do not want more than k parties winning, since the number of goods are fixed (giving out more ASICs interferes with the total mining rate in the system) and hence, in the decision phase, k bidders are declared winners (after breaking ties). We want the auction protocol to satisfy the following properties:

Bid confidentiality. No information about the bid and the bidder's identity is revealed during the bidding stage.

Bid independence. A bidder cannot create a bid which depends on a submitted honest bid. For example, a bid which is one higher than a submitted bid.

Public verifiability. The outcome of the auction can be verified publicly.

Censorship-resilience. No party is prevented from submitting a bid in the auction.

Binding (unretractability). A winning bidder cannot retract the bid in the decision stage or abort the auction.

We do not consider other auction designs in this work, and leave exploring other economic models and auctions to future work.

5.1 All-or-Nothing Broadcast

Towards implementing an auction mechanism on chain for ASICs, we define and construct a building block that allows "encryption to the future on the blockchain". At a high level, this building block immediately give a way to create an all-or-nothing broadcast channel. Such a broadcast channel has two phases. In the first phase, parties can input messages, but no one can retrieve them. After some delay, parties can retrieve all the messages in the second phase. The adversary can abort the delivery of all messages, but it cannot prevent the delivery of a specific message.

Functionality $\mathcal{F}_{\textsf{AON-BC}}$

The all-or-nothing broadcast parametrised by times τ_1, τ_2, τ_3.
The function maintains a buffer B which initially is empty.

Phase 1: Input

- This phase starts at time τ_1 and ends at time τ_2.
- During this phase parties can input messages which are then stored in buffer B.

Phase 2: Output

- At time τ_3 the functionality outputs the content of buffer B to the adversary.
- The functionality then asks the adversary if the content of B should be released to the honest parties.
- If the adversary released the contents the honest parties can now query the functionality for the content of buffer B.

Adversarial Access

The adversary can at any point in time query the size of the buffer B (but not its content before Phase 2).

We use this property of the AON broadcast channel in Section 5.2 to create a simple auction. In the first part, parties input their bids into the broadcast and in the second phase we use the now revealed bids to determine the winner of the auction.

We believe all-or-nothing broadcast and encryption to the future on the blockchain could be of independent interest and useful for other applications.

All-or-nothing broadcast on the blockchain. The following protocol uses the encryption-to-the-future idea to construct an all-or-nothing broadcast based on an ATE scheme. It runs in three phases which are described below. We assume that parties have access to a $(\mathfrak{t}, \mathfrak{n})$- ad hoc threshold-encryption scheme parametrized by the number of key-shares $\mathfrak{n}$ of which $\mathfrak{t}$ are required to decrypt.

Protocol All-or-nothing broadcast

The protocol is parametrized by the threshold encryption parameters $(\mathfrak{t}, \mathfrak{n})$ and delays d_1, d_2, d_3.

Phase 0: Key Generation

- This phase lasts until $\mathfrak{n}$ blocks have been created.
- If a miner creates a block, they use $\textsf{KeyGen}$ to generate key shares and add the public-key share to the block.
- After $\mathfrak{n}$ blocks have been created, any party can obtain the set of all public keys $\textsf{PK} = \{\textsf{pk}_i\}$.
- The miners have to keep the secret keys secret until the Decryption Phase.

Phase 1: Encryption

- This phase starts right after the key-generation phase and lasts until d_1 blocks have been created.
- To encrypt a message m to the future, a party encrypts m under $\textsf{PK}$ using $\textsf{Enc}$ and posts the resulting ciphertext c on the blockchain.

Phase 2: Decryption

- This phase starts d_2 blocks after the encryption phase and lasts d_3 blocks.

- Any miners that created a block in the key-generation phase publishes their secret key (on the blockchain).
- Once $\mathfrak{t}+1$ secret keys have been published, all messages can be decrypted (publicly) using Dec.

Lemma 2. *Let $\mathfrak{n}$ be so large, that in any $\mathfrak{n}$ consecutive blocks at most $\mathfrak{t}$ blocks were created by dishonest parties. Let d_1 be so large, that at least one block in any d_1 consecutive blocks was created by an honest party. Then, the above protocol achieves the all-or-nothing broadcast $\mathcal{F}_{\mathsf{AON\text{-}BC}}$.*

Proof. By the assumption on $\mathfrak{n}$ and $\mathfrak{t}$, at most $\mathfrak{t}$ blocks in Phase 0 are created by dishonest parties. Thus the adversary knows at most $\mathfrak{t}$ keys and can therefore not decrypt messages which have been put on the blockchain before the honest parties release their keys in Phase 2. This means that the adversary can only see the number of messages, i.e. the size of buffer B. The adversary can also not prevent honest parties from inputing messages, as we assume that in Phase 1 at least one honest block is created which could contain all honest inputs.

In Phase 2, the honest parties from Phase 0 will all release their generated keys. This allows the adversary to read all inputed message, i.e. the contents of B. Now, if the adversary also releases enough of its keys, the honest parties can read all the inputed messages.

The time τ_1 corresponds to the time it takes to create $\mathfrak{n}$ blocks, time τ_2 is the time it takes to create $n + d_1$ blocks, and τ_3 is the time it takes to create $\mathfrak{n} + d_1 + d_2$ blocks.

Lemma 3. *Assuming an honest majority of Virtual ASICs, there exist parmeter $\mathfrak{n}$, $\mathfrak{t}$, and d_1 such that the above protocol achieves $\mathcal{F}_{\mathsf{AON\text{-}BC}}$ on the virtual ASICs blockchain.*

Proof. By Lemma 1, the blockchain satisfies the chain-quality property. This means that for any $\mathfrak{n}$ (large enough), there is a bound $\mu < \mathfrak{n}$ on the number of dishonest blocks within $\mathfrak{n}$ consecutive blocks. We can set $\mathfrak{t} = \mu$. Furthermore, chain-quality also guarantees that for d_1 large enough there must be at least one honest block within d_1 consecutive blocks. The statement follows by Lemma 2.

The bound on the number of dishonest blocks within $\mathfrak{n}$ consecutive blocks in the above proof depends on the size of $\mathfrak{n}$ and the fraction of honest ASICs. If we assume a strong honest majority, e.g. $\frac{2}{3}$ honest ASICs, $\mu < \frac{\mathfrak{n}}{2}$ holds even for smaller $\mathfrak{n}$. In this case, the protocol achieves the stronger delay broadcast $\mathcal{F}_{\text{D-BC}}$. This broadcast has the same functionality as $\mathcal{F}_{\mathsf{AON\text{-}BC}}$ except that the adversary can no longer abort (i.e. we have guaranteed output delivery).

Lemma 4. *If $\frac{2}{3}$ of all virtual ASICs are under honest control, then for large enough $\mathfrak{n}$ the above protocol achieves $\mathcal{F}_{\text{D-BC}}$ on the virtual ASICs blockchain for $\mathfrak{t} = \mathfrak{n}/2$.*

Proof. A large honest majority of virtual ASICs (e.g. $\frac{2}{3}$) ensures that there exists $\mathfrak{n}$ such that less than $\mathfrak{t} := \frac{\mathfrak{n}}{2}$ of $\mathfrak{n}$ consecutive blocks are dishonest. There also exists d_1 (resp. d_3) such that any d_1 (resp. d_3) consecutive blocks contain at least one honest block. By Lemma 2 we get the properties of $\mathcal{F}_{\mathsf{AON\text{-}BC}}$. Furthermore, in Phase 2 all honest parties will release their keys which is enough to decrypt all inputs. The adversary cannot prevent that key release as there will be at least one honest block in Phase 2. Thus the adversary cannot abort the channel.

On disincentivizing aborts. The above protocol might only achieve $\mathcal{F}_{\mathsf{AON\text{-}BC}}$ either due to high corruption or choice of parameters (e.g. a $\mathfrak{t} > \frac{n}{2}$ which allows for smaller $\mathfrak{n}$). In this case the dishonest parties can abort the channel.

We can use financial rewards to disincentivize aborts, and achieve stronger properties of the delay broadcast. One option is that block rewards are only paid out if miners release their key shares within the decryption phase. To prevent early-release, if someone posts the key share of a miner before the decryption phase they get a fraction (e.g. $\frac{1}{10}$) of the miners block reward while the miner gets nothing. If the key-share is released too late, no one gets a reward. Note that this solution still requires d_1 large enough to prevent input censorship.

If block rewards do not provide enough incentives, one can additionally require that miners need to pay a certain amount of stake into escrow. If they misbehave, in particular if they do not release their keys in time, these funds get slashed. The value of the slash fund will depend on the actual incentive parties have for not releasing a key.

An alternative to prevent aborts is using time-lock puzzles as seen in [DDM+18]. Roughly, the idea is that miners in Phase 0 not only publish the public-key, but also a time-lock puzzle containing the secret key and a zero-knowledge proof showing that the correct value is inside the puzzle. If a key is not released in Phase 2, parties simply solve the time-lock puzzle to retrieve the key. This way the adversary could at most delay the channel, but not abort.

5.2 The Auction Protocol

We now show how to implement a censorship-resilient ASIC auction using the broadcast functionality from the previous section. We assume a system that supports *shielded transactions*; transactions that provide privacy of the sender, receiver and the amount being transfered. A $(k+1)$st price auction protocol is described below where k ASICs are won. In the case that there are less than k bids in an auction, we assume there is a reserve price paid by the winner(s).

To ensure that the auction is binding, parties are required to prepay their bidding amount onto an escrow account. A bid is a tuple

$$(\mathsf{vk}, \mathsf{amount}, \pi, \mathsf{pk})$$

consisting of bidders' signature verification key vk, the bidding amount, the proof of payment π, and a VRF public-key pk. The bid is signed under by the bidder. The proof of payment shows that the bidder prepaid amount to the escrow account. It is essentially the secrets that open the shielded transaction, together with reference to the payment transaction. The VRF key pk will be used to create a new ASIC in the event that the bidder wins.

Protocol ASIC Auction

Bidding Phase To submit a bid a party P does the following.

1. The party generates a fresh VRF key pair, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}$.
2. The party makes a shielded transaction of the bid amount to a pre-specified escrow address. The transaction includes a return address.
3. Next, the party inputs their signed bid $(\mathsf{vk}, \mathsf{amount}, \pi, \mathsf{pk})$ into the broadcast channel.

Decision Phase

1. All bids are read from the broadcast channel.
2. Invalid bids (e.g. that are not accompanied by a valid proof of payment) are discarded.
3. The valid bids are sorted in decreasing order of bid value into a list. The first k valid bids (after breaking ties) are announced as winners. The $(k+1)$ valid bid amount is announced as the auction price.
4. Each winning bid defines a new ASIC $(\mathsf{id}, \mathsf{vk}, \mathsf{pk}, \mathsf{initMR}, j_0)$ where vk, pk are the signature verification key and the VRF public key of the bid. The $\mathsf{id}, \mathsf{initMR}, j_0$ are determined by the auction itself, vk, pk by the winning party's input to the broadcast channel.
5. A transaction is made from the auction address to each winning return address containing the difference amount between the bid made and the amount paid, and the ASIC attributes. Transactions returning the bid amounts are made for all other non-winning participants.
6. If the auction fails, participants reveal their bid transactions. The bid amount is returned to the return addresses for all auction participants.

Remark 1. The IDs of the newly created ASICs could for example be $i.j$ where i is the number of the auction, and j is a number between 1 and k. The initial mining rate of the ASICs will depend on the economic model used. The spawn epoch of the ASICs must be at least the epoch after the auction's epoch.

Remark 2. In a system without full-fledged shielded addresses, we can hide payments to the auction address among other payments as follows: Consider, for instance, an address that are n-bit hashes of a public key like in Bitcoin. To bid in the auction, a user can make a payment with a transaction to an address that is computed as the hash of the phase number of the auction, together with a secret. Then, to provide a proof of valid payment in the AON-BC channel, the user reveals the secret and a reference to this transaction, along with a return address. This proves that the transaction is a well-formed bid payment.

Lemma 5. *The protocol* ASIC Auction *achieves desired properties when broadcast channel is* $\mathcal{F}_{\text{D-BC}}$.

Proof. Bid confidentiality: The shielded transaction to the escrow address does not reveal the amount nor the bidder's identity. The properties of the broadcast channel ensure that the actual bid cannot be extracted before the decision phase. *Bid independence:* This also follows directly from the properties of shielded transactions and the broadcast. *Public verifiability.* A bid contains all information necessary to verify its validity. In particular, the proof of payment allows to check that the bidder paid the right amount to the escrow. The consistency of the blockchain and the broadcast imply that all parties will agree on the set of (valid) bids. *Censorship-resilience.* The use of shielded transactions prevents the adversary from censoring honest payments to the escrow account. The adversary cannot prevent parties from input to the broadcast nor prevent the overall output of the broadcast.

Binding (unretractability). First note that the adversary cannot prevent bids from being revealed by broadcast. Furthermore, any bid which is deemed valid in the decision phase is linked to a valid payment to the escrow account. This winner of the auction therefore already paid for their ASICs.

Corollary 1. *Assuming* $\frac{2}{3}$ *of all virtual ASICs are under honest control, the protocol* ASIC Auction *using the protocol from Section 5.1 to emulate broadcast achieves desired properties.*

This follows directly from Lemmata 4 and 5.

If we only use the all-or-nothing broadcast for our auction protocol, the adversary can publicly abort the auction. That is, the adversary can look at the bids and then decide to abort by not releasing them from the broadcast channel.

Lemma 6. *The protocol* ASIC Auction *achieves auction with abort when broadcast channel is* $\mathcal{F}_{\mathsf{AON\text{-}BC}}$. *All other desired properties still hold.*

Corollary 2. *Assuming an honest majority of virtual ASICs the protocol* ASIC Auction *using the protocol from Section 5.1 to emulate broadcast achieves auction with abort.*

We note that in our setting (assuming rational parties) we have the means to deal with aborts and punish the adversary for doing so (see Section 5.1). Thus achieving an auction with abort is fine.

5.3 Other uses For Encryption to the Future

The encrypt to future functionality can be extremely useful for other interactions on the blockchain. One example of this is as a solution to the front-running problem which is often experienced in Ethereum [DGK+19]. Allowing traders to commit to trades simultaneously via our AON broadcast functionality and then execute all trades together in a batch would solve the problem. We note that similar suggestions have been raised with respect to conventional trading in the stock exchange [BCS15], where flash trades appear to also hinder exchanges.

6 The Properties of Virtual ASICs

In this section we discuss the design space that virtual ASICs offer, and provide examples for the various choices available to the protocol designer. The different choices solely affect the economics and incentives that underlie mining. This joins other economic concerns such as setting the money creation rate (e.g., Bitcoin's halving of the block rewards every roughly 4 years).

Unhindered by physical aspects that may affect the ASICs behavior, we may still want to mimic some of their properties even in our virtualized setting. We detail some properties below, along with the rationale for including them. These are only selected examples. There are many other dimensions that virtual ASICs can be modified under (including changes to the auction mechanism, or the rate at which ASICs themselves are created)

ASIC supply. To avoid having a single party buy enough virtual ASICs to attack the protocol, we suggest that ASICs should be created at a limited rate. A constant rate seems to be a simple and natural choice. As we have suggested in Section 5, new ASICs can be auctioned as a way to discover the correct price that participants place upon their value. Additional options may include creating ASICs exactly as needed to replace one that have expired, or allowing an unlimited amount to be sold at every epoch, but at an increasing marginal price.

The profits from ASIC sales or other costs can be either burned or returned to the miners much later in the form of block rewards.

Powering ASICs. In addition to the costs of obtaining hardware, conventional ASICs require electricity in order to run. This translates to a cost on the ASIC owner that is paid as the ASIC is powered. An important feature is that ASICs can be turned off – relinquishing any potential rewards, but also avoiding paying for electricity. Indeed, miners are known to have turned off their ASICs after rewards have dropped sharply (due to a halving of the reward in Bitcoin or due to price fluctuations). It has been argued that the miner's ability to decide whether or not to power its ASIC effectively makes the ASIC equivalent to a bundle of European options [YZ20]. This makes powering the ASIC equivalent to exercising a European option at its maturity date.

We therefore discussed in Section 3.2 the means by which virtual ASICs can be powered. Each ASIC owner pays in advance for powering the ASIC. We stress that payments for virtual electricity do not constitute a waste that affects the environment. No actual resources are consumed. In fact, payments changing hands do not affect the overall social welfare – whilst the paying party does lose utility, the recipients of funds gains. Even if money is burned, overall welfare remains the same as burning funds raises the value of the currency held by others in the long run and is therefore functionally equivalent to a transfer to these individuals.

One important aspect in which virtual ASICs differ from physical ones is in the way electricity costs relate to the value of the cryptocurrency. In physical mining, electricity costs are external to the blockchain and do not fluctuate with the currency that is being mined. In our setting, the lack of reliable external exchange rate information (a pricing oracle) prevents us from pegging the electricity price to some stable external asset. It would be interesting to explore ways of integrating oracles or stable coins into this pricing mechanism. We leave such aspects to future work.

We also note that the electricity pricing of virtual ASICs has the advantage of being independent of the miner's geographic location. This prevents centralization due to low electricity costs in certain regions of the world.

Rent & maintenance costs. Owners of physical ASICs incur additional constant costs that do not go away if ASICs are turned off. For example, rent must be paid to house the ASICs in some facility. This can be mimicked in our setting as well, in the form of periodic payments. If an ASIC owner does not pay, the ASIC can be lost or degraded. Having such an ongoing constant cost is beneficial to the security of the blockchain as it ensures that parties that hold ASICs are active and are involved in mining. They cannot hoard inactive ASICs for long periods of time without incurring a high cost (such dark mining power is always a threat as it can "wake up" and be used to attack the system).

Technically, the rent payment can be approximated by making un-powered ASICs degrade faster. Alternatively, we can add a rental payment mechanism similar to the power transactions as seen in Section 3.2. The degradation will then also depend on the rental payment, e.g. if no rent is paid the ASIC might lose all its mining rate.

Degrading mining rates over time. Another aspect of physical hardware is wear and tear. Our virtual ASICs can be set to degrade as well, whether in how fast they mine or in how much power is needed to run them. Technically, this is achieved by degradation function as seen in Section 3.2. Moreover, the cost of electricity can also depend on the age of the ASIC.

The decay of ASICs (along with the creation of new ASICs to replace them) may be desirable as a mechanism to encourage and allow for participation by newcomers.

Technological improvement. Another aspect that current mining hardware exhibits is that newer hardware tends to be faster and more efficient in terms of electricity. Since the probability of succeeding in mining is relative to the total available mining power, making newer ASICs more potent than older ones is not effectively different from degrading older ASICs.

To model this effect for virtual ASICs one can auction off more powerful ASICs over time or use the degradation mechanism.

6.1 Extremes of the Design Space

Proof of stake. One extreme end of the design space is to allow Virtual ASICs to run with no power or rent costs at all. In this case, the miner pays only the initial costs of the hardware. Here, each ASIC can be seen as a token that allows to participate in the proof of stake lottery.

Having ASICs created at a constant rate, with a slow rate of decay in their mining rating (this corresponds to inflation) will allow miners to be replaced slowly while still tying miners to the system – Their ASIC yields returns over long periods of time and therefore the well-being of the system should still be aligned with their own interests.

Proof of burn. At the other extreme, we can imagine a Virtual ASIC that is completely free (or very cheap), that is sold at unlimited quantities. The only costs to pay are those required to power it. In this case, the system is effectively allowing those who burn more money at every epoch to mine more. We do not recommend using this extreme method, as it may allow malicious miners to increase their mining rate disproportionately for short periods of time.

7 Conclusion and Future Work

In this work, we introduced *Virtual ASICs* as a generalization of proof-of-stake. We showed how to base a consensus protocol on Virtual ASICs, and a meachnism to to sell them in sealed-bid auctions on-chain. Along the way, we introduce a novel all-or-nothing broadcast functionality in blockchains that is of independent interest. We leave for future work exploring the design space of virtual ASICs in the context of economic models and incentivising behavior. It is also interesting to explore *privacy-preserving* consensus based on virtual ASICs. We believe that the auction can be made private, while using a public list of ASICs by having the bid input to the broadcast channel include freshly generated signature/VRF keys together with a zero-knowledge proof of valid payment to the escrow account. Then, a private lottery in a spirit similar to [GOT19, KKKZ19, BMSZ20] can be made to work where the winners publish blocks containing a proof that a particular ASIC won the lottery.

References

[BCS15] Eric Budish, Peter Cramton, and John Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015.

[BGK$^+$18] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. Cryptology ePrint Archive, Report 2018/378, 2018. `https://eprint.iacr.org/2018/378`.

[BGM16] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *International Conference on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.

[bit11] Proof of stake instead of proof of work. `https://bitcointalk.org/index.php?topic=27787.0`, July 2011.

[BLMR14] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

[BMSZ20] Foteini Baldimtsi, Varun Madathil, Alessandra Scafuro, and Linfeng Zhou. Anonymous lottery in the proof-of-stake setting. Cryptology ePrint Archive, Report 2020/533, 2020. `https://eprint.iacr.org/2020/533`.

[Bon16] Joseph Bonneau. Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.

[BPS16] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016. `http://eprint.iacr.org/2016/919`.

[CDE$^+$16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[DDM$^+$18] Dominic Deuber, Nico Döttling, Bernardo Magri, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Minting mechanisms for blockchain – or – moving from cryptoassets to cryptocurrencies. Cryptology ePrint Archive, Report 2018/1110, 2018. `https://eprint.iacr.org/2018/1110`.

[DGK$^+$19] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234*, 2019.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 66–98. Springer, Heidelberg, April / May 2018. `doi:10.1007/978-3-319-78375-8_3`.

[DHMR07] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. CCA2-secure threshold broadcast encryption with shorter ciphertexts. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 35–50. Springer, Heidelberg, November 2007.

[DHMR08] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Ad-hoc threshold broadcast encryption with shorter ciphertexts. *Electronic Notes in Theoretical Computer Science*, 192(2):3–15, 2008.

[DY20] Ivan Damgård and Sophia Yakoubov. Bounds on ad hoc threshold encryption. Cryptology ePrint Archive, Report 2020/618, 2020. `https://eprint.iacr.org/2020/618`.

[Eth] Ethash. URL: `https://eth.wiki/en/concepts/ethash/ethash`.

[GHM$^+$17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. `http://eprint.iacr.org/2017/454`.

[GOT19] Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 690–719. Springer, Heidelberg, May 2019. `doi:10.1007/978-3-030-17653-2_23`.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 466–485. Springer, Heidelberg, May / June 2010. `doi:10.1007/978-3-642-13190-5_24`.

[KKKZ19] Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019. `doi:10.1109/SP.2019.00063`.

[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. IEEE Computer Society Press, May 2016. `doi:10.1109/SP.2016.55`.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, Heidelberg, August 2017. `doi:10.1007/978-3-319-63688-7_12`.

[MHWK16] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution*, pages 1–6, 2016.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 620–649. Springer, Heidelberg, August 2019. `doi:10.1007/978-3-030-26948-7_22`.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[OM14] Karl J. O'Dwyer and David Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 280–285, 2014.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996.

[RSY18] Leonid Reyzin, Adam Smith, and Sophia Yakoubov. Turning hate into love: Homomorphic ad hoc threshold encryption for scalable mpc. Cryptology ePrint Archive, Report 2018/997, 2018. `https://eprint.iacr.org/2018/997`.

[RT06] Michael O Rabin and Christopher Thorpe. Time-lapse cryptography. 2006.

[TSE19] Itay Tsabary, Alexander Spiegelman, and Ittay Eyal. Just enough security: Reducing proof-of-work ecological footprint. *arXiv preprint arXiv:1911.04124*, 2019.

[YZ20] Aviv Yaish and Aviv Zohar. Pricing asics for cryptocurrency mining. *CoRR*, abs/2002.11064, 2020. URL: `https://arxiv.org/abs/2002.11064`, `arXiv:2002.11064`.