This work was published as

A. Hartl, T. Zseby and J. Fabini, "BeaconBlocks: Augmenting Proof-of-Stake with On-Chain Time Synchronization," *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 353-360.

DOI: 10.1109/Blockchain.2019.00055

BeaconBlocks: Augmenting Proof-of-Stake with On-Chain Time Synchronization

Alexander Hartl
Institute of Telecommunications
TU Wien
Vienna, Austria
alexander.hartl@tuwien.ac.at

Tanja Zseby
Institute of Telecommunications
TU Wien
Vienna, Austria
tanja.zseby@tuwien.ac.at

Joachim Fabini
Institute of Telecommunications
TU Wien
Vienna, Austria
joachim.fabini@tuwien.ac.at

***Abstract*—Blockchain protocols based on Proof-of-Stake (PoS) algorithms aim to provide an alternative to the energy-consuming Proof-of-Work mining procedure. Following a PoS algorithm, nodes have to agree on the miner next eligible to contribute a block and on the point in time he is allowed to broadcast it. The latter requirement raises to the need for synchronous clocks.**

In this paper we describe BeaconBlocks, a new scheme for constructing PoS protocols. A major difference to former work is incorporating time synchronization as an essential element of the protocol itself, gaining independence of the nodes' clocks and allowing the protocol to resist attacks on clock synchronization infrastructure. To this end, we describe both a mechanism for obtaining the correct time during node startup and for retaining synchronicity of estimated time during a node's lifetime.

In contrast to prior work, our approach for miner selection exhibits an interleaved unslotted structure. We show that fairness is achieved when miners follow our scheme and we provide a discussion of attack possibilities, allowing developers to choose secure parameters when adopting the scheme.

***Index Terms*—blockchain, time synchronization, proof-of-stake**

I. Introduction

Bitcoin [1] achieves consensus in a truly elegant way. Performing a technique known as Proof-of-Work (PoW), network nodes solve cryptopuzzles to authorize block creation, ideally making influence on the network proportional to the amount of capital each participant is willing to invest.

The huge demand for electric power that comes along with PoW is the source of heated debates. Not only does it cause the environmental balance of any protocol based on PoW to be catastrophic, it also leads to substantial amounts of money leaving the network in form of electricity costs, eventually increasing the costs for everyone using it.

It therefore comes as no surprise that soon after the invention of bitcoin, protocol designers were looking for ways to combine the advantages blockchains yield with less energy-consuming consensus techniques. The most well-known approach is Proof-of-Stake (PoS).

PoS in the classical definition means that influence on the network each participant has, measured as the targeted number of blocks this participant is permitted to contribute, is made proportional to the amount of capital he currently holds on the blockchain. To formulate our findings in the most general framework possible, however, we consider PoS in a more general sense. Hence, we regard PoS as the targeted number of contributed blocks being derived using some determinate function from previous data on the blockchain. This definition generalizes the above definition of classical PoS but allows a variety of other methods, which might be beneficial in various respects. Due to the generality of this definition, we adopt the term *miner* for referring to a node that is actively contributing blocks, despite its usual use in PoW protocols. Accordingly, we term the targeted number of contributed blocks for a miner, relative to the total block count, his associated *mining power*.

In this work, we describe a new scheme for obtaining PoS and investigate its security. To this end, we investigate several attack scenarios and discuss their impact on security, allowing proper protocol parametrization in practical deployments.

Chain-based PoS [2], as described in this paper, is a stochastic consensus technique and is also known as eventual-consensus PoS [3]. In essence, it can be considered a form of repeated decentralized lottery, the winners being rewarded by being permitted to create blocks. Unlike recent approaches like [4, 5] we use an unslotted approach loosely comparable to the Exp-algorithm described in [6]. Given a certain recent block, every miner draws a number pseudorandomly, which serves as delay that is imposed on the broadcasting time of this miner's block. Hence, the miner with the lowest random number wins by having his block accepted first by the network. In contrast to former methods, our scheme exhibits an interleaved structure, yielding security benefits compared to methods as used in [6].

The major novel idea we present in this paper, however, is performing chain and time synchronization in a joint manner, thus avoiding the need for external mechanisms to keep an accurate clock throughout the mining process.

Due to the basic functioning of PoS protocols, a common notion of time is crucial for secure protocol operation. However, currently used clock synchronization techniques have shown to yield substantial drawbacks in terms of security [7]. Depending on how strong an adversary's incentives are, even the signal from GPS satellites cannot be trusted as reliable source of time information [8]. Google's Roughtime protocol[1] comes closest to the security properties we desire when building a blockchain protocol on top of it, but also fails to

[1] https://roughtime.googlesource.com/roughtime

provide the desired degree of decentralization. Indeed, due to its stochastic functioning, chain-based PoS can achieve a substantial degree of decentralization. Basing chain-based PoS on traditional clock synchronization, querying just a few centralized time servers, would thus severely harm decentralization and thereby the protocol's security.

Hence, in this paper we propose a mechanism for integrating time synchronization in the blockchain protocol itself. The main mechanism we use for this purpose retains synchronicity of time during a node's lifetime. An administrator is expected to give his client software an estimate of current time during software startup and the client remains synchronized ever after using the network, much like grid-synchronized clocks remain synchronized using the electric power grid.

Providing a correct timestamp when powering up appears to be a substantially easier task than guaranteeing a correct clock throughout the whole lifetime of the mining process. Yet, to cover a wide range of application scenarios, we finally also describe a mechanism for obtaining time information during the startup phase if no reliable clock is available.

II. Related Work

The idea of relying on stake distribution on the blockchain itself to eliminate the energy-consuming mining process was born soon after the invention of bitcoin [1] and several PoS protocols [9, 10, 11, 12] were implemented. For these early approaches a large number of attacks has been found [13, 14], allowing the protocols to be attacked with even small amounts of stake. Recently, important work has been done on analyzing blockchain security and improving it [15, 16, 17, 18, 3]. At the same time, PoS protocols like [4] and [5] emerged in the academic community, providing important new concepts and extensive security evaluations in the framework of Universal Composability (UC).

PoS methods mentioned above can all be subsumed under the class of chain-based [2] or eventual-consensus [3] PoS. Apart from these methods, blockchain protocols deploying the older theory of Byzantine Fault Tolerance (BFT) were developed [19, 20, 21, 22, 23], requiring tradeoffs known as the CAP theorem [24].

In this paper we focus on time synchronization. Almost all protocols devised so far either assume synchronized clocks to be given relying, e.g., on external clock synchronization protocols, or transition to a different notion of time which is agreed upon using (different) consensus techniques.

An exception worth mentioning is the NEM [12] blockchain which features built-in clock synchronization by performing NTP-like synchronization procedures against randomly selected network peers. This clock synchronization procedure can be considered a consensus procedure on its own, which is executed before the actual chain synchronization takes place. For assessing the protocol's security, both mechanisms have to be considered. In particular, unlike our scheme, being performed independently, clock synchronization in this case has no access to the Sybil attack prevention which is inherent to blockchains.

TABLE I
Most important notation.

$t \in \mathbb{R}$	Real time.
$\tau_{\text{RTC}}(t) \in \mathbb{R}$	A node's real-time clock.
$\tau_{\text{Mon}}(t) \in \mathbb{R}$	A node's monotonic clock.
$\delta \in \mathbb{R}$	A node's clock offset.
$\tilde{\tau}(t) \in \mathbb{R}$	A node's estimated time ($\tilde{\tau}(t) = \tau_{\text{Mon}}(t) + \delta$).
$\sigma(\mathcal{B}) \in \mathbb{R}$	Chain time at block $\mathcal{B}$.
$T_B \in \mathbb{R}^+$	Average inter-block interval.
$2\omega \in \mathbb{N}$	Stake delay.
$\Delta \in \mathbb{R}^+$	Time correction per block.
$\mathcal{B}$	A block.
ρ	Public key identifying a miner.
$q_\rho(\mathcal{B}) \in [0,1]$	ρ's mining power after processing block $\mathcal{B}$.
$H(\cdot)$	Cryptographic hash function with output length L_H.
S	Random seed for miner selection.
R	Randomness obtained by signing S.
k_s/k_p	Private/public key of the node executing Algorithm 1.

Interesting in the present context is also the work by Fan et al. [25] who proposed a clock synchronization mechanism secured by blockchain technology, particularly for use in the Internet of Things. Compared to their approach, we do not rely on a functioning blockchain to build on to achieve clock synchronization, but instead integrate time synchronization as a fundamental element of our PoS algorithm. However, a technique like [25] might be used to extend our method to increase accuracy of the time synchronization mechanism.

While borrowing some ideas from work mentioned above, our method has major differences to all of them. It is composed of the following components:

1) On-chain time synchronization: We introduce mechanisms to recover time information from the blockchain itself, obtaining a PoS algorithm without the need for external clock synchronization and allowing the protocol to resist attacks on clock synchronization infrastructure.

2) Interleaved unslotted PoS: Unlike prior work, we deploy a new miner selection mechanism which uses neither the concept of epochs nor that of timeslots. We describe how miner selection works and show that fairness is achieved when miners follow the protocol's rules.

III. Miner Selection: An informal description

Blockchain protocols counter major attacks by making influence on the network proportional to "something expensive". Even though this statement is oversimplified, "something expensive" can be identified as computational power in the case of PoW and as on-chain capital in the case of classical PoS. Influence on the network, on the other hand, can be quantified by the number of blocks one is able to create in a given period of time or, in other words, by the probability of being permitted to contribute a block at a specific time.

Constructing a PoS protocol therefore essentially means constructing a repeated decentralized lottery scheme, which satisfies predetermined winning probabilities for miners.

Digital signatures constitute a major component of our scheme. We therefore directly identify a miner by his public key ρ. Similar to previous work [4, 19], we require the used digital signature scheme to be unique, ensuring that a

miner can find exactly one signature string for a given signed message. We use a cryptographic hash function H, producing an output string of length L_H bits and assume a defined encoding, so that the output of H can directly be interpreted as an integer in $1, \ldots, 2^{L_H}$. Finally, S denotes a random seed for miner selection varying over time (see Section V for details).

For classical PoS, the number of blocks, a miner is allowed to contribute, is determined by relative on-chain capital after the processing of a certain block. Hence, if K_ρ denotes ρ's capital after the processing of block $\mathcal{B}$, and M denotes the set of all miners, the quotient $q_\rho(\mathcal{B}) = K_\rho / \sum_{\tilde{\rho} \in M} K_{\tilde{\rho}}$ is used for determining ρ's probability of contributing a block at a certain later point. However, taking a closer look, for the functioning of a PoS method it is irrelevant if this stake ratio, or any other function of data on the blockchain, is what is used for this purpose. In the following, we will therefore assume a function $q_\rho(\mathcal{B}) \in [0, 1]$, so that $\sum_{\rho \in M} q_\rho(\mathcal{B}) = 1$, to be given, and refer to $q_\rho(\mathcal{B})$ with the neutral term "mining power".

In the light of the above considerations, the task of our scheme has to be to dictate when miners are eligible to contribute blocks. This time is specified relative to a recent block. Hence, for every block, each miner is assigned a *lag time*, which has to pass before being allowed to create a block, pseudorandomly. For reasons which will become evident presently, ρ's lag time will be obtained proportional to the value $-\frac{1}{q_\rho(\mathcal{B})} \ln(2^{-L_H} H(R))$, where R is a signature for seed S under public key ρ. By incorporating a signing operation, each miner obtains a different time for creating his block.

The prefactor $1/q_\rho(\mathcal{B})$ has an intuitive appeal. Indeed, the larger the mining power, the sooner a block will be accepted by the network. The remaining part $-\ln(2^{-L_H} H(R))$, on the other hand, is used for generating exponentially distributed random numbers with mean 1.

A peculiarity about our method is not only allowing the block with the lowest lag time to extend the chain, but potentially also blocks arriving later, linking them one after the other. The result is an interleaved structure as depicted in Fig. 1. In this figure, both miner 1 and miner 2 are able to extend the blockchain based on block *A*. For both miners different lag times are computed, after which they are able to broadcast blocks *C* and *F*, respectively. We will say throughout this paper that blocks *C* and *F* *stem from* block *A*. In this example, we allow both miners to extend the chain subsequently. The figure also shows that due to interleaving, neither *C* nor *F* is the first to extend the blockchain after *A*, but a block *B* stemming from a block preceding *A*.

A final issue, which has to be considered, is as follows. Facing a potentially unlimited number of miners, the approach as described above would lead us to the ill-fated situation of

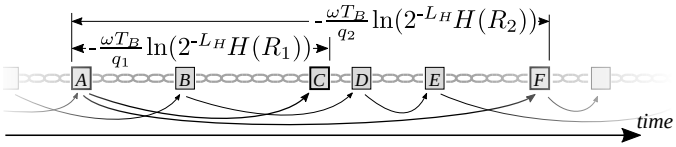


Fig. 1. An exemplary block sequence.

the block rate rising exponentially with time. We therefore constrain the lag time, measured as number of blocks having arrived meanwhile, by a network parameter $\omega \in \mathbb{N}$ and, to obtain an average inter-block interval of T_B, scale the lag time to $-\frac{\omega T_B}{q_\rho(\mathcal{B})} \ln(2^{-L_H} H(R))$. In the example of Fig. 1, *F* is then declined for $\omega < 5$.

As a consequence of interleaving, mining power $q_\rho(\mathcal{B})$ in any block might be used in any of ω consecutive blocks. It shall be emphasized, however, that our method is not epoch-based in the sense of other recent algorithms like, e.g., [4].

Note, furthermore, that while the interleaved structure is used for determining the order in which miners are eligible to create blocks, the order of transactions is given by the resulting linear order of blocks, which is illustrated as a chain-shaped line in Fig. 1. In the example of this figure, block *F* must not include transactions spending outputs that were already spent in block *E* or preceding blocks.

IV. The Clock in PoS

Trust assumptions for blockchain protocols differ significantly from traditional systems. A time synchronization mechanism based on a PoS blockchain, hence functioning under the same trust assumptions, has important use cases:

1) Unlike PoW, in PoS there is no natural mechanism limiting block creation. Hence, we artificially have to introduce delays for broadcasting blocks, giving rise to the necessity of a common clock.
   As a consequence, the clock becomes crucial for being able to correctly select the best chain available. When performing time synchronization based on a blockchain as described in this paper, hence, both proper chain synchronization depends on time synchronization and time synchronization depends on chain synchronization, requiring a joint synchronization mechanism.
2) Certain types of transactions and smart contracts executed on blockchains depend on a common clock. In particular, most current cryptocurrencies allow transactions to be prevented from execution before a certain date and time. Hence, a common clock is necessary to make all network nodes agree on the set of transactions that cannot yet be included in the blockchain. For example, such transactions are of fundamental importance for techniques like the lightning network[2].
3) A clock synchronization technique that works under the same trust assumptions as blockchains, and does not rely on trusted centralized servers, might be used for various purposes. For example, decentralized clock synchronization might integrate into the usual clock synchronization infrastructure gaining immunity against attacks on time servers or on unauthenticated NTP messages.
   In this paper we do not focus on this application scenario. Adoption of the technique to meet requirements of such applications with respect to, e.g., accuracy, is thus left for further research.

[2]https://lightning.network/

An adversary's inability to arbitrarily influence a node's time is of fundamental importance for our application. However, it is important to note that (1) and (2) have substantially lower accuracy requirements than many traditional clock synchronization scenarios, NTP or PTP are used for.

In particular, for (1) the clock is used as a reference to compare the arrival times of blocks to. Deviating clocks, hence, have the same effect as increased block propagation delays. Since the network has to be designed to withstand the impact of block propagation to a certain extent, clock deviations that are well below the network's propagation delays do not have a significant impact on the network.

A similar argument holds for (2). Since transactions can only be processed by being included in blocks, precise timing for when a transaction has to be accepted is not possible. Hence, clock deviations that are well below the average inter-block interval have no significant impact.

These very specific requirements justify a crude approach for performing time synchronization in our case: Disregarding any propagation delays of blocks, we can use blocks as beacons that are seen by all network nodes at approximately the same time and, hence, allow the nodes to deduce time information from received blocks. To this end, every network node is equipped with a monotonic clock $\tau_{\text{Mon}}(t) \in \mathbb{R}$ and a real-time clock $\tau_{\text{RTC}}(t) \in \mathbb{R}$, where $t \in \mathbb{R}$ denotes the real physical time as it is displayed on a wall clock.

The monotonic clock $\tau_{\text{Mon}}(t)$ is guaranteed to advance monotonically with time, but does not necessarily represent wall clock time, i.e. there is a variable offset $\delta \in \mathbb{R}$, so that $\tau_{\text{Mon}}(t)+\delta = t$. As will be described in Section V, we assume the absolute values of the monotonic clocks' relative clock drifts to be upper-bounded by a certain value. Our scheme works by estimating δ. For the sake of notational brevity, we abbreviate a node's estimated time as $\tilde{\tau}(t) = \tau_{\text{Mon}}(t) + \delta$.

In addition to $\tau_{\text{Mon}}(t)$, every network node has access to a real-time clock $\tau_{\text{RTC}}(t)$ which can be used to query an estimate of the wall clock time t. We do not make any assumptions about the accuracy of a node's real-time clock, but we assume that the mining power-weighted median of the real-time clocks of all nodes gives a reasonably good estimate of t.

Like all chain-based PoS blockchains, our scheme works by imposing delays on block broadcasting, where delays can be deduced from data on the blockchain. The sum of all imposed delays therefore provides an estimate of the time that has passed since the blockchain's launch date, a network parameter every node is aware of. Hence, by summing together all imposed delays up to the current block, we additionally obtain a "chain time" $\sigma \in \mathbb{R}$.

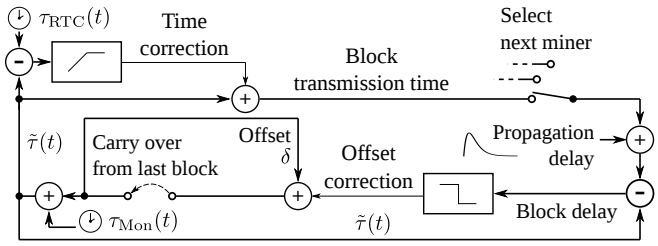


Fig. 3. Interaction of clocks.

When a block is received, we would thus have in an ideal world $\sigma = \tilde{\tau}(t) = \tau_{\text{RTC}}(t) = t$. Our scheme adjusts both the offset δ and block broadcasting times to approach equality of these different time sources. By correcting δ only by a small amount for each block, we can prevent large fluctuations.

In more detail, when receiving a block, a node compares $\tilde{\tau}(t)$ to σ, the time it expected the block to arrive at. A block arriving early can be interpreted as $\tilde{\tau}(t)$ being too low. Hence, the node slightly increases δ when a block arrives early. Similarly, the node slightly decreases δ when the block arrives late. Since every node sees the new block at roughly the same time, the network becomes self-synchronizing in the sense that every node has approximately the same time $\tilde{\tau}(t) \approx \sigma$.

However, without further measures, $\tilde{\tau}(t)$ would drift away from the real time t. Hence, when creating a new block, nodes also have to compare $\tilde{\tau}(t)$ to $\tau_{\text{RTC}}(t)$ and appropriately adjust the time they broadcast the block. For example, when $\tilde{\tau}(t) < \tau_{\text{RTC}}(t)$, new blocks are advanced. Nodes thus receive the blocks early and increase their δ, eventually reducing the deviation between $\tilde{\tau}(t)$ and $\tau_{\text{RTC}}(t)$.

Fig. 3 depicts the interactions between a node's clocks. Starting from an offset value δ from the last block in the middle of the figure, we obtain the estimated time $\tilde{\tau}(t) = \tau_{\text{Mon}}(t) + \delta$. The transmission time of the next block is found by using $\tilde{\tau}(t)$ as reference. However, this transmission time is adjusted based on the deviation of $\tilde{\tau}(t)$ to $\tau_{\text{RTC}}(t)$. The next miner is then determined as described in the previous Section III and his block is received after having been delayed by a random propagation delay. Compared to $\tilde{\tau}(t)$, the block thus has a positive or negative block delay. This delay is used to compute a correction value for the offset δ, which will be used for the processing of the next block.

Fig. 2 shows how the time synchronization process proceeds

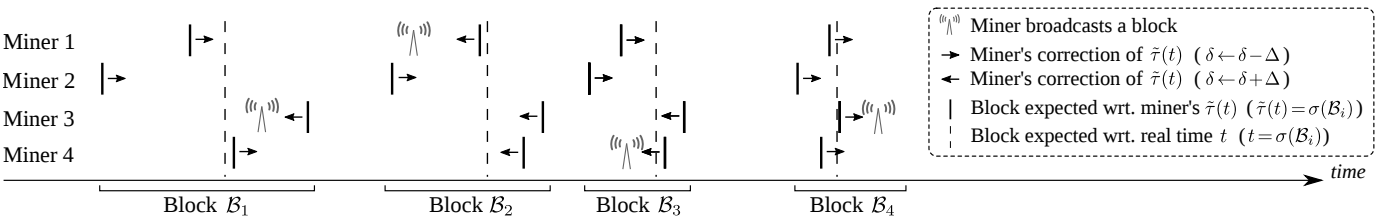


Fig. 2. An example of the time synchronization process.

Algorithm 1 Processing a block $\mathcal{B} = (h, D, a, R, \rho, s)$.

1: assert $(H(\mathcal{B}^{-1}) = h) \wedge (D \text{ extends } \mathcal{B}^{-1}) \wedge (1 \leq a \leq \omega) \wedge \mathsf{verify}_{\rho}(S(\mathcal{B}^{-a}), R) \wedge \mathsf{verify}_{\rho}(h \mid D \mid a \mid R \mid \rho, s)$
2: set $\delta(\mathcal{B}) \leftarrow \delta(\mathcal{B}^{-1}) - \Delta \cdot \text{sgn}\left(\tau_{\text{Mon}}(t) + \delta(\mathcal{B}^{-1}) - \sigma(\mathcal{B})\right)$
3: wait until $\tau_{\text{Mon}}(t) \geq \sigma(\mathcal{B}) - \delta(\mathcal{B}^{-1})$
4: UPDATETIP($\mathcal{B}$)
5: **with** $(\hat{a}, \hat{R}, \hat{\sigma}) \leftarrow \arg\min_{(\hat{a},\hat{R},\hat{\sigma}) \in P(\mathcal{B})} \{\hat{\sigma} \mid \hat{\sigma} > \sigma(\mathcal{B})\}$ **do**
6: wait until $\tau_{\text{Mon}}(t) \geq \hat{\sigma} - \delta(\mathcal{B}) + \min(2\Delta, \tau_{\text{Mon}}(t) + \delta(\mathcal{B}) - \tau_{\text{RTC}}(t))$
7: **if** $\mathcal{T} = \mathcal{B}$ **then**
8: Collect transactions into $\hat{D}$ and broadcast $\left(H(\mathcal{B}), \hat{D}, \hat{a}, \hat{R}, k_p, \mathsf{sign}_{k_s}(H(\mathcal{B}) \mid \hat{D} \mid \hat{a} \mid \hat{R} \mid k_p)\right)$
9: **end if**
10: **end with**

at the example of four consecutive blocks $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$. The vertical bars show when miners expect a block to arrive based on their time $\tilde{\tau}(t)$, i.e. when $\tilde{\tau}(t) = \sigma(\mathcal{B}_i)$. The vertical dashed line shows when the arrival of blocks is expected based on the real time t, i.e. when $t = \sigma(\mathcal{B}_i)$. Finally, the arrows indicate the time correction miners undertake when they receive a block and the symbol $\overset{((\cdot))}{\Vert}$ shows the time blocks are broadcast.

Primarily, a miner uses his clock $\tilde{\tau}(t)$ for finding the right time to broadcast a block. However, to reduce deviations from real time, each miner also corrects this time based on his $\tau_{\text{RTC}}(t)$. As shown for miner 1 in block $\mathcal{B}_2$, this does not always lead to a correction in the right direction, as miner 1 in this case has an inaccurate real-time clock.

Nevertheless, by the synchronization mechanisms described in this section, deviations are reduced gradually. While the miners' times $\tilde{\tau}(t)$ differ substantially at block $\mathcal{B}_1$, both the deviation of miners' times to each other and the deviations to real time t have assumed an acceptable level for block $\mathcal{B}_4$.

V. BEACONBLOCKS

We now present BeaconBlocks, a scheme that combines interleaved miner selection as described in Section III with on-chain time synchronization as described in the previous Section IV. The scheme has three parameters:

- $T_B \in \mathbb{R}^+$ is the target average interval between any two consecutive blocks, e.g. $T_B = 60s$.
- A *stake delay* 2ω with $\omega \in \mathbb{N}$ which is, very informally stated, the minimum time, measured as number of blocks, before an investment in mining power can yield revenues, e.g. $\omega = 2160$.
- A time correction $\Delta \in \mathbb{R}^+$, so that Δ / T_B is an upper bound for the absolute values of the nodes' relative clock drifts, e.g. $\Delta = 0.6s$.

BeaconBlocks proceeds by broadcasting blocks $\mathcal{B} = (h, D, a, R, \rho, s)$. We denote $\mathcal{B}$'s ith predecessor in the chain as $\mathcal{B}^{-i}$ and by $\mathcal{B}_h, \mathcal{B}_D, \mathcal{B}_a, \mathcal{B}_R, \mathcal{B}_\rho$ and $\mathcal{B}_s$ the corresponding fields in $\mathcal{B}$.

A block consists of h, the hash of the previous block, D, denoting transaction data, $a \in \mathbb{N}$, specifying the block $\mathcal{B}^{-a}$, $\mathcal{B}$ stems from, R, a pseudorandom value the block's lag time is derived from, ρ, the block creator's public key and s, a signature authenticating the block. Similar to Section III, we denote by $q_\rho(\mathcal{B})$ ρ's mining power after processing block $\mathcal{B}$, e.g. ρ's relative capital in the case of classical PoS.

Chain-based PoS can be considered a repeated decentralized lottery, where the winner is rewarded by being permitted to create one block. For this pseudorandom miner selection we thus need a random seed S. Preventing manipulations of this seed value has turned out to be a tricky task. We here use ideas of [4, 5]. In particular, in accordance to [5] the seed value $S(\mathcal{B})$, relevant for blocks stemming from block $\mathcal{B}$, is determined by hashing together the randomness values of all blocks preceding $\mathcal{B}^{-\omega}$, i.e.

$$S(\mathcal{B}) = H\left(S(\mathcal{B}^{-1}) \mid \mathcal{B}_R^{-\omega}\right).$$

From the seed we compute lag times for block broadcasting as $-\frac{\omega T_B}{q_\rho(\mathcal{B}^{-2\omega})} \ln(2^{-L_H} H(R))$, where R is a signature for $S(\mathcal{B})$ under ρ. Hence, when receiving a block $\mathcal{B}$, which stems from block $\mathcal{B}^{-\mathcal{B}_a}$, we can compute the time when $\mathcal{B}$ was expected to arrive at as

$$\sigma(\mathcal{B}) = \sigma(\mathcal{B}^{-\mathcal{B}_a}) - \frac{\omega T_B}{q_{\mathcal{B}_\rho}(\mathcal{B}^{-\mathcal{B}_a - 2\omega})} \ln\left(2^{-L_H} H(\mathcal{B}_R)\right).$$

On the other hand, when creating a block extending the chain at a block $\mathcal{B}$, it can stem from any of the last ω blocks. Hence, we obtain a pool of possible times for when a miner is able to create blocks, which can be written as the set

$$P(\mathcal{B}) = \left\{ \left(a+1, R, \sigma(\mathcal{B}^{-a}) - \frac{\omega T_B}{q_{k_p}(\mathcal{B}^{-a-2\omega})} \ln(2^{-L_H} H(R)) \right) \;\middle|\; 0 \leq a < \omega \wedge R = \mathsf{sign}_{k_s}(S(\mathcal{B}^{-a})) \right\}.$$

With these definitions of $\sigma(\mathcal{B})$ and $P(\mathcal{B})$, we can now write the algorithm for processing a block as depicted in Algorithm 1 and Algorithm 2. Receiving a new block $\mathcal{B}$, in the first line of Algorithm 1 a node verifies the formal requirements of the block, i.e. validity of signatures and of transactions extending the chain. In line 2 it then adapts the offset δ. Here, $\text{sgn}(\cdot)$ denotes the signum function. If the block arrived later than expected, δ is reduced by Δ, if it arrived earlier, it is increased by Δ. If the block arrived early, the node then waits for the block's expected time to arrive and decides in line 4 if the block constitutes the new tip of the currently best chain.

This decision is described in Algorithm 2. To prevent long-range attacks, as will be described in Section VII, we use an

approach similar to [4] which only considers the first ω blocks after a fork to make the decision about the best chain available. Hence, in Algorithm 2 we first find $a_{\mathcal{T}} \in \mathbb{N}_0$ and $a_{\mathcal{B}} \in \mathbb{N}_0$ so that $\mathcal{B}^{-a_{\mathcal{B}}} = \mathcal{T}^{-a_{\mathcal{T}}}$ is the last common predecessor of $\mathcal{B}$ and the current tip $\mathcal{T}$. We then distinguish if the fork dates back more than ω blocks. If this is the case, the node selects the chain which is better in the first ω blocks after the fork, considering the blocks' expected arrival times σ (see Fig. 5). Otherwise, it deploys a simple longest-chain rule.

In line 5 of Algorithm 1, a miner then determines when he is permitted to create the next block, and in line 6 waits for this time to come. In line 6 also $\tau_{\text{RTC}}(t)$ is queried and the block's transmission time is corrected by $\tau_{\text{Mon}}(t) + \delta(\mathcal{B}) - \tau_{\text{RTC}}(t)$ to reduce deviations of σ from the real time. This correction is tightly upper-bounded as a miner might otherwise put his chance to create a block at risk.

Finally, when the time for creating a block has come, a new block is broadcast in line 8, given that $\mathcal{B}$ is still the tip of the best chain, rerunning Algorithm 1 for the new block.

VI. Fairness in an Honest World

At the heart of a chain-based PoS blockchain we face the problem of selecting miners for block creation in a fair manner. In most settings, the revenues, a miner gains from contributing in the mining process, can be well approximated as being proportional to the number of blocks this miner contributes. Hence, in the present context we can understand fairness as the property that a miner who holds a mining power of q in some block, is able to contribute q blocks on average. In this section, we investigate if this property is achieved if miners follow the algorithm described in the previous Section V.

Due to the setting in which most public blockchains operate, we have to allow miners to split their mining power to an arbitrary number of accounts. We furthermore make the assumption that miners aim to maximize the number of blocks they contribute. Hence, we assume a miner ρ who wants to extend the current chain at the most recent block $\mathcal{B}$ and possessed a mining power of $q_\rho(\mathcal{B}^{-a})$ in some recent block $\mathcal{B}^{-a}$ with $a < \omega$, which dates back a time $v_0 = \sigma(\mathcal{B}) - \sigma(\mathcal{B}^{-a})$. For the sake of notational brevity, we will abbreviate ρ's mining power as $q = q_\rho(\mathcal{B}^{-a})$.

Deriving the lag times as described in the previous sections from the logarithm of independent uniformly distributed random variables, the lag times for individual signature evaluations constitute independent exponentially distributed random variables with mean $\omega T_B/q$.

Let us denote by $F(v) \in [0,1]$ the probability of being able to extend the chain within a time of $v \in \mathbb{R}^+$. For one signature evaluation, the probability of the corresponding lag time to lie inside $[v_0, v_0 + v]$ can be found from the cumulative distribution function (cdf) of an exponentially distributed random variable with mean $\omega T_B/q$ as

$$F(v) = \exp\left(-\frac{qv_0}{\omega T_B}\right) - \exp\left(-\frac{q(v_0+v)}{\omega T_B}\right).$$

Performing a first-order Taylor approximation, observe that for $q \sim 0$, $F(v) \sim qv/\omega T_B$, becoming exact for $q \to 0$. Hence, when evenly splitting his q to $L \to \infty$ accounts, and again computing the probability of any of the lag times to lie inside $[v_0, v_0 + v]$, ρ can at his will instead obtain

$$\tilde{F}(v) = 1 - \left(1 - \frac{q}{L}\frac{v}{\omega T_B}\right)^L \xrightarrow{L\to\infty} 1 - \exp\left(-\frac{qv}{\omega T_B}\right)$$

in place of $F(v)$. Note that $F(v) = \tilde{F}(v)$ for $v_0 = 0$. Since, however, $\frac{\partial}{\partial v_0}F(v) \leq 0$, we know additionally that $F(v) < \tilde{F}(v)\,\forall\, v \in \mathbb{R}^+$. This inequality means that, independent of when other miners are able to create blocks, chances that ρ's block comes first are always better when splitting q to multiple accounts. Hence, assuming rational actors, accounts *will* always be split if $F(v)$ deviates significantly from $\tilde{F}(v)$.

$\tilde{F}(v)$ can thus be interpreted as the cdf of the inter-block interval of any miner who held mining power q in a recent block. Inter-block intervals become exponentially distributed with mean $\omega T_B/q$ and, by the properties of the exponential distribution, inter-block intervals of the chain, any set of miners $\tilde{M}$ creates, become exponentially distributed with mean $\omega T_B/(\sum_{a=0}^{\omega-1}\sum_{\tilde{\rho}\in\tilde{M}} q_{\tilde{\rho}}(\mathcal{B}^{-a}))$.

In particular, having a mining power q in one recent block, as assumed above, the delay of the earliest block, ρ is able to create, is exponentially distributed with mean $\omega T_B/q$ and the delay of the earliest block, *anyone else* is able to create, is exponentially distributed with mean $\omega T_B/(\omega - q)$. The probability of ρ to be first to create a block can be computed as $\int_0^\infty \frac{q}{\omega T_B}\exp(-\frac{qv}{\omega T_B})\exp(-(1-\frac{q}{\omega})\frac{v}{T_B})dv = q/\omega$. Since mining power in one block can be used for a total of ω consecutive blocks, we obtain an expected number of contributed blocks for ρ of $\omega\frac{q}{\omega} = q$. This is what we wanted to prove.

Algorithm 2 Deciding between two chains.

```
1: function UPDATETIP(B)
2:     with (a_T, a_B) ← arg min_{a_T, a_B ∈ N_0} {a_T a_B | T^{-a_T} = B^{-a_B}} do
3:         if min(a_T, a_B) > ω then
4:             set T ← arg min_{i ∈ {T,B}} {σ(i^{-a_i + ω})}
5:         else if a_B > a_T then
6:             set T ← B
7:         end if
8:     end with
9: end function
```

VII. Attacks

In this section we analyze several attack scenarios on BeaconBlocks. We model the adversary as possessing a certain static mining power $q \in [0, 0.5]$. The adversary is thus able to contribute blocks according to the rules the protocol determines, but might manipulate their broadcasting times or the chains the blocks extend, to gain an advantage. We assume, in particular, that the adversary does not possess secret keys from legitimate nodes, or equivalently, we consider the mining power associated with compromised keys as part of the adversary's mining power.

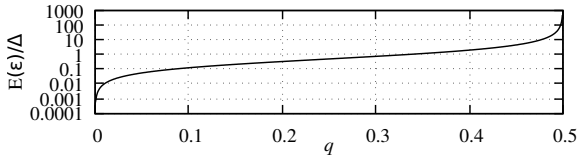


Fig. 4. Expected time error in presence of an adversary.

In the real world, an adversary might also make use of propagation delays between legitimate nodes or even cause additional delays. Since an in-depth discussion of such attacks is beyond the scope of this paper, we refer the reader to existing work on blockchain security (e.g. [18]) and remark additionally that we expect the time synchronization mechanism to double the impact of substantial propagation delays on our scheme's security in the worst case.

A. Exploiting time

Considering the joint synchronization of chain and time, it comes as no surprise that an adversary can exploit the time synchronization mechanism to gain a benefit when trying to fork the blockchain. An adversary of mining power q can broadcast a block every T_B/q time units on average. Transmitting all these blocks early, the time on the adversary's fork will run faster, as the offset is increased with every block.

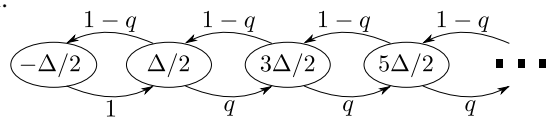
After a time interval T, the time on the adversary's fork will be off by $\Delta T/\frac{T_B}{q}$. Hence, in this time interval the adversary will be able to create $(T + T\frac{\Delta}{T_B}q)/\frac{T_B}{q}$ blocks. Our adversary thus is able to create a chain that is longer than it is supposed to be by a factor of $1 + \frac{\Delta}{T_B}q$ and, hence, gains an effective mining power increase by the same factor.

Assuming a (very large) upper bound for relative clock drifts of, e.g., $\frac{\Delta}{T_B} = 1\%$, an adversary can gain 50% effective mining power with a mining power of 49.75%. The impact of this attack strategy is thus very limited or even entirely negligible.

B. On time

As a side effect of our scheme, nodes obtain a common time $\tilde{\tau}(t) \approx t$. While it is not absolutely necessary for secure protocol operation, considering the discussion in Section IV, it is desirable to have this common time tamper-proof against an adversary. We now analyze the error $\epsilon = \tilde{\tau}(t) - t$, an adversary of mining power q can cause. Since we in this section are only interested in deviations an adversary might cause and not in random deviations due to propagation delays or inaccurate real-time clocks, we assume that in normal protocol operation ϵ constantly fluctuates between $-\frac{\Delta}{2}$ and $\frac{\Delta}{2}$.

Aiming to make the error ϵ as large as possible, the adversary will broadcast every block early, increasing nodes' δ by Δ. Honest miners, on the other hand, will notice a large error and broadcast blocks late, reducing δ by Δ. Altogether, ϵ then transitions between the discrete states $\{-\frac{\Delta}{2}, \frac{\Delta}{2}, \frac{3\Delta}{2}, \ldots\}$ whose transition probabilities can be expressed as a Markov chain:



Steady-state probabilities for this Markov chain resolve to

$$\Pr\left\{\epsilon = -\frac{\Delta}{2}\right\} = \frac{1-2q}{2(1-q)} \quad \text{and}$$

$$\Pr\left\{\epsilon = \frac{\Delta}{2} + i\Delta\right\} = \frac{1-2q}{2(1-q)^2}\left(\frac{q}{1-q}\right)^i \quad \forall\, i \in \mathbb{N}_0.$$

Computing the expectation over this distribution, we obtain an expected error $\mathrm{E}(\epsilon) = \Delta\frac{q}{1-2q}$, which is shown in Fig. 4. As visible from the figure, also in this case the adversary's influence is very limited as long as his mining power does not approach majority.

C. Long-range attacks

Long-range attacks aim to fork the chain at a block dating back a substantial period of time, possibly several weeks or even months. Producing such a long fork, an adversary might be able to craft a chain which appears to have been generated by a larger mining power than is actually possessed by the adversary. For an overview of major long-range attacks, the reader is referred to [3].

In BeaconBlocks we, similarly to [5], take seed values relevant for block $\mathcal{B}$ from block $\mathcal{B}^{-\omega}$ and mining power from block $\mathcal{B}^{-2\omega}$. With this approach, long-range attack scenarios look as depicted in Fig. 5. While the adversary might be able to produce a fork which is longer than the legitimate chain, he cannot modify mining power or seed values for the first ω blocks in the fork and is therefore unable to improve his fork in the first ω blocks. Similar to [4], Algorithm 2 therefore only considers the first ω blocks after a fork for chain selection, an approach which is known as *strict chain density statistics* [3].

A downside of this approach is that ω has to be chosen large enough, so that a chain of length ω yields a reasonably good estimate of the mining power of the nodes that have created the chain. As the blockchain is created by stochastic mechanisms, an adversarial chain of length ω might be better than the legitimate chain just by chance.

We thus have to analyze the probability of the sum of the intervals between ω consecutive blocks in the adversarial fork to be smaller than the sum of intervals between ω consecutive blocks in the legitimate chain. With large enough ω, we can approximate the sum of independent exponentially distributed random variables with mean T_B/q to be distributed according to $\mathcal{N}(\omega T_B/q, \omega T_B^2/q^2)$. Thereby, the probability of an adversary with mining power q to create a better chain than legitimate miners with mining power $1-q$ can be computed as $Q\left(\sqrt{\omega}\frac{q^{-1}-(1-q)^{-1}}{\sqrt{q^{-2}+(1-q)^{-2}}}\right)$. If we, for example, target a probability of this attack scenario of $Q(\ldots) < 10^{-10}$ with $q = 45\%$, we obtain $\omega > 2044$.

Finally, checkpointing [3] might be added as an additional defense mechanism, effective against *posterior corruption*.

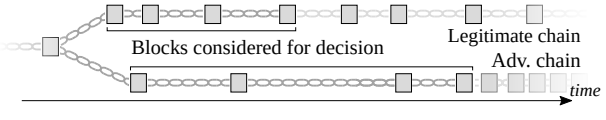


Fig. 5. Adversarial long-range forks of the blockchain.

VIII. Timeless initial synchronization

The previous sections described a mechanism for retaining synchronicity of the nodes' time $\tilde{\tau}(t)$ throughout the mining process. Hence, we assumed an administrator to be able to provide the correct time for $\tilde{\tau}(t)$ during software startup. In fact, in most cases this assumption is likely to be achievable. Furthermore, the initial synchronization phase is not critical for the consensus procedure and, hence, every node can choose a method that suits his needs best. Yet, in some cases it is desirable to synchronize without any reliable time information, e.g. if the system should be able to (re)boot autonomously. Astonishingly, this is indeed possible.

To see this, we note that the pure existence of a block $\mathcal{B}$ can be interpreted as a vote for the block's time $\sigma(\mathcal{B})$ having passed already. Thus, if we are offered a chain seemingly coming from the future, which, however, has an estimated mining power of more than a certain threshold $\hat{q} \in [0.5, 1]$, we can accept the majority vote for our offset being too low.

Following this idea, for performing a timeless initial synchronization, a node starts by setting $\tilde{\tau}(t)$ to the blockchain's launch time. As long as in a time window $T \in \mathbb{R}^+$ more than $\hat{q}T/T_B$ blocks indicate a too low δ, the node increases δ, eventually approaching $\tilde{\tau}(t) \approx t$, so that time synchronization as discussed in the previous sections can take over.

The method introduces the parameters $\hat{q} \in [0.5, 1]$ and $T \in \mathbb{R}^+$, which, however, are not global network parameters, but can be tuned by each user separately. For the process to be secure, T must be chosen large enough to ensure that an adversary with mining power q is unable to create more than $\hat{q}T/T_B$ blocks in a time interval T. The number of blocks, the adversary is able to create in an interval T, obeys a Poisson distribution with mean qT/T_B. Targeting a probability of 10^{-10} of the adversary to be able to create more than $\hat{q}T/T_B$ blocks, we obtain, e.g., $T/T_B > 71$ for $\hat{q} = 80\%, q = 30\%$.

Before being able to set $\tilde{\tau}(t)$ close to the real time t, a node has to await the creation of $\hat{q}T/T_B$ blocks. The drawback of this method is thus that the synchronization phase during software startup is prolonged significantly.

IX. Conclusions

In this paper we showed relations between PoS and secure decentralized time synchronization. We presented BeaconBlocks, a scheme for obtaining chain-based PoS that includes time synchronization as a major element and deploys a new interleaved unslotted approach for miner selection. We described procedures for both obtaining time information when performing initial synchronization on node startup and for retaining synchronicity later throughout the mining process.

Many attack possibilities have to be considered when designing a PoS blockchain. We therefore investigated several attacks and showed how algorithm parameters influence the impact on security, that different attacks have.

We brought up the research question if a variant of BeaconBlocks could replace traditional clock synchronization. A scheme meeting even higher accuracy requirements would enable exciting new application areas of blockchain technology.

References

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

[2] V. Buterin *et al.*, "Proof of stake FAQ," *Ethereum Wiki*, Mar 2019. [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[3] P. Gai, A. Kiayias, and A. Russell, "Stake-bleeding attacks on proof-of-stake blockchains," in *Proc. CVCBT'18*, June 2018, pp. 85–92.

[4] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," Cryptology ePrint Archive, Report 2018/378, 2018, https://eprint.iacr.org/2018/378.

[5] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake," Cryptology ePrint Archive, Report 2016/919, 2017, https://eprint.iacr.org/2016/919.

[6] S. Popov, "A probabilistic analysis of the nxt forging algorithm," *Ledger*, vol. 1, pp. 69–83, 2016.

[7] R. Annessi, J. Fabini, F. Iglesias, and T. Zseby, "Encryption is futile: Delay attacks on high-precision clock synchronization," 2018. [Online]. Available: http://arxiv.org/abs/1811.08569

[8] "Study maps 'extensive russian gps spoofing'," *BBC*, Apr 2019. [Online]. Available: https://www.bbc.com/news/technology-47786248

[9] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2017. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[10] "Whitepaper:nxt." [Online]. Available: https://nxtwiki.org/wiki/Whitepaper:Nxt

[11] L. Goodman, "Tezos – a self-amending crypto-ledger," 2014. [Online]. Available: https://www.tezos.com/static/papers/white_paper.pdf

[12] "Nem – technical reference," 2018. [Online]. Available: https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf

[13] A. Poelstra, "On stake and consensus," 2015.

[14] N. Houy, "It will cost you nothing to'kill'a proof-of-stake crypto-currency," 2014.

[15] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," 2013. [Online]. Available: http://arxiv.org/abs/1311.0243

[16] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Proc. EUROCRYPT'15*, 2015, pp. 281–310.

[17] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proc. PODC'17*, 2017, pp. 315–324.

[18] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Proc. FC'15*, 2015, pp. 507–527.

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. SOSP'17*, 2017, pp. 51–68.

[20] J. Kwon, "Tendermint: Consensus without mining," 2014.

[21] D. Schwartz, N. Youngs, A. Britto *et al.*, "The ripple protocol consensus algorithm," 2014. [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf

[22] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," 2015. [Online]. Available: https://www.stellar.org/papers/stellar-consensus-protocol.pdf

[23] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 2017. [Online]. Available: http://arxiv.org/abs/1710.09437

[24] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, pp. 51–59, Jun. 2002.

[25] K. Fan, S. Wang, Y. Ren, K. Yang, Z. Yan, H. Li, and Y. Yang, "Blockchain-based secure time protection scheme in IoT," *IEEE Internet of Things Journal*, 2018.