On-Chain Timestamps Are Accurate

Apostolos Tzinas[2,3], Srivatsan Sridhar[1], and Dionysis Zindros[1,3]

[1] Stanford University
[2] National Technical University of Athens
[3] Common Prefix

Aug 30, 2023
Last update: October 24, 2023

Abstract. When Satoshi Nakamoto introduced Bitcoin, a central tenet was that the blockchain functions as a *timestamping server*. In the Ethereum era, smart contracts widely assume on-chain timestamps are mostly accurate. In this paper, we prove this is indeed the case, namely that recorded timestamps do not wildly deviate from real-world time, a property we call *timeliness*. Assuming a global clock, we prove that all popular mechanisms for constructing blockchains (proof-of-work, longest chain proof-of-stake, and quorum-based proof-of-stake) are *timely* under honest majority, but a synchronous network is a necessary condition. Next we show that all timely blockchains can be suitably modified, in a black-box fashion, such that all honest parties output exactly the same ledgers at the same round, achieving a property we call *supersafety*, which may be of independent interest. Conversely, we also show that supersafety implies (perfect) timeliness, completing the circle.

1 Introduction

In the original Bitcoin paper [16], Nakamoto refers to Bitcoin as a *distributed timestamping server*. Chain timestamps have long been assumed to correspond to real-world time by the community. DeFi applications with more than $1B in monthly volume rely [8] on the precision of these recorded timestamps. They assume new transactions are not only included soon — a well-understood property known as *liveness* — but also that their on-chain recorded timestamps are not far in the past compared to real time. It is indeed the case that, in practical blockchain deployments, honest nodes only accept blocks with timestamps up to a short window into the future[4]. However, an adversary can produce a block with a fabricated past timestamp, and can still cause this block to be accepted by the honest nodes retroactively by, for example, cooking up a lucky slightly longer Bitcoin chain. In this paper, we show that these fabrications cannot exceed a certain *timeliness* limit, which we calculate.

We put forth the notion of *timeliness*, formalizing the folklore understanding that timestamps recorded on-chain cannot deviate arbitrarily from real world time. In particular, we define a timeliness parameter v which bounds how far in

[4]For example, this is part of the Bitcoin [17] and Ethereum [21] implementations.

the past a timestamp can be. Under a synchronous network, a global clock, and honest majority, we prove this virtue materializes in all three popular flavors of permissionless blockchains, namely proof-of-work, longest chain proof-of-stake, and quorum-based proof-of-stake, and analytically calculate their timeliness parameter.

Blockchain systems executed in networks with delay, even when assuming a global clock and bounded delay, allow honest parties to reach consensus, but the ledgers reported by honest parties are typically merely prefixes of one another. Under perfectly synchronized clocks, we show that any *timely* blockchain can be modified to achieve *exactly equal* ledgers at every point in time, a property we call *supersafety*, albeit with the introduction of a small confirmation delay.

The introduction of these two properties in this abstract fashion simplifies the proofs of security of protocols built on top. For example, cross-chain constructions such as Babylon [18] are simpler to prove if on-chain timestamps are used instead of block heights, but timeliness is needed to do this. Consensus constructions which require the synchronized production of randomness, such as Ouroboros [6, 14], can also benefit by simpler security arguments from the abstraction of the supersafety property.

Our contributions. In this paper, we make the following contributions:

1. We introduce the notion of ledger *timeliness*. We prove that three exemplary cases of blockchain protocols (Bitcoin, Streamlet, and Ouroboros) are timely and calculate their timeliness(Section 4). We also show that network synchrony is required for timeliness (Section 5).
2. We build a *supersafe* protocol from any timely ledger. Supersafe protocols allow parties with synchronized clocks to reach the exact same conclusion about their ledgers at the exact same time. Conversely, we reduce from supersafety back to timeliness, illustrating that the two properties are morally equivalent.

Overview of the results. An overview of the results of this paper is illustrated in Figure 1. Our first result is proving that all popular ways of constructing distributed ledgers are, indeed, timely, if we assume a global clock and a synchronous network. The timeliness follows from an intermediate technical property we call *freshness* of chains, which we prove for three protocols Streamlet, Bitcoin Backbone, and Ouroboros, that are representatives of the three major classes of protocols: quorum-based proof-of-stake, longest chain proof-of-work, and longest chain proof-of-stake respectively (Section 4). The Streamlet variant is proven in the partially synchronous setting after GST. We show that timeliness before GST is impossible for protocols that are always-safe and live after GST (Section 5). In practice, the assumption of a global clock can be relaxed by tolerating an additional deviation in the timestamps equal to the maximum clock deviation.

Next, we perform two simple but instructive black-box reductions that are, perhaps, somewhat unexpected: The first reduction is from timeliness to supersafety (Section 6). In this reduction, we delay the reported ledger until all

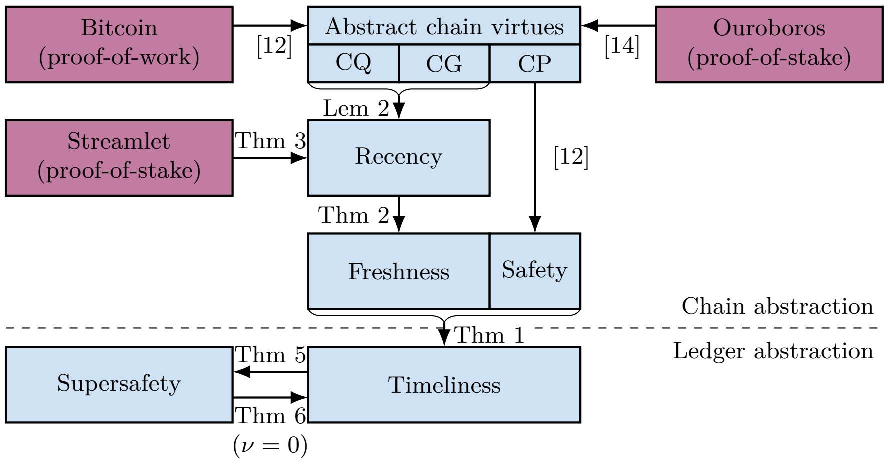

Fig. 1: Paper overview. Bitcoin and Ouroboros attain the abstract chain virtues of Common Prefix, Chain Quality, and Chain Growth (top) [12, 14]. We show that CQ and CG imply the intermediate property of recency (Lemma 2, middle). Recency is also achieved by Streamlet (Theorem 3, middle) and is sufficient to prove freshness (Theorem 2, middle). Freshness, together with safety, is sufficient to prove timeliness (Theorem 1). In safe and live protocols, from timeliness we can achieve supersafety and vice versa (Theorems 5-6, bottom).

transactions have timestamps enough into the past to ensure all other parties have seen them. This allows us to achieve a remarkable property: All honest parties report the exact same ledgers at the exact same time. In this reduction, parties who desire supersafety for their application can individually choose to obtain supersafety at the cost of additional confirmation delay, while other parties continue to have faster confirmation and standard safety. The second reduction is from supersafety back to (perfect) timeliness (Section 6.2). To do this, every online node reports the new transactions they see with their current timestamp, and due to supersafety, these are the same. Some additional work is needed to allow intermittently online nodes to catch up, which we also show.

Related work. Arguments of chain timeliness appear at the heart of various security proofs in the blockchain literature. In the variable difficulty Bitcoin Backbone paper [13], it is shown that the different honest parties performing target recalculation on different chains must arrive at roughly the same result, relying on roughly synchronized timestamps. In GearBox [7], a sharding protocol, shards timestamp messages on a "control chain" and such timestamps are used to determine whether a shard is live. To do this, the authors of [7] define the notion of a timed ledger (same as our notion of a temporal ledger), and conjecture that Hotstuff, Bitcoin, and Ouroboros are timely. We prove these facts. In Babylon [18], Bitcoin is used as a secure timestamping service to provide additional security to proof-of-stake blockchains. In the Ouroboros protocol [14], the randomness production from epoch-to-epoch requires chopping off a suitable number of *slots*, achieving a form of supersafety. We posit that the abstraction of timeliness and supersafety into stand-alone properties can aid in simplifying the security proofs of existing and future protocols. The blockchain itself has also been used [1, 4] as a mechanism for clock synchronization among parties with desynchronized and drifting clocks, a long-standing problem [15]. In our work, we show that, if the clocks of honest parties are already synchronized, possibly by using [1, 4], on-chain reported timestamps roughly correspond to real-world time.

Previously, the community was confused about whether on-chain timestamps are accurate. In some discussions, it was incorrectly claimed[5] that all accepted block timestamps must be within the same tolerance that honest nodes use to accept blocks, ignoring the possibility of fabricated timestamps in blocks produced by the adversary. On the other hand, others conjectured that on-chain timestamps can be vastly inaccurate stating that *timestamps can differ radically from the actual time, outside the network* [20]. In this work, we resolve this confusion, derive the timeliness parameter and rigorously show the conditions under which it holds. Supersafety, although never stated formally, was achieved since the early days of consensus by Dolev and Strong [9]. However, contrary to us, their construction does not support intermittently online (sleepy) clients, and was designed for a different era of consensus, with a much smaller scale than today's systems in mind, poor performance, and no support for dynamic availability.

2 Preliminaries & Model

Notation. We use $[\ ]$ to denote the empty sequence. We denote by $|C|$ the length of sequence C, by $C[i]$ the i^{th} (zero-based) element of sequence C, and by $C[-i]$ the i^{th} element from the end. We use $C[i{:}j]$ to mean the subsequence of C from the i^{th} element (zero-based inclusive) to the j^{th} element (zero-based exclusive). Omitting i takes the sequence to the beginning, and omitting j takes the sequence to the end. By $C_1 || C_2$, we mean the concatenation of sequences C_1 and C_2. We use the set notation $B \in C$ to iterate through a sequence C. We use the set builder notation $[B \in C : p(B)]$ to filter the elements of sequence C that satisfy p. We write $C_1 \preccurlyeq C_2$ when C_1 is a prefix of C_2 and $C_1 \prec C_2$ to mean that the prefix is strict. We write $C_1 \sim C_2$ when $C_1 \preccurlyeq C_2 \lor C_2 \preccurlyeq C_1$. We use κ to denote the security parameter.

Model. Execution occurs in discrete rounds $1, 2, \ldots$, of polynomial number in the security parameter κ. The protocol is executed by parties of two types: n *nodes* and an unlimited number of *clients*. The adversary controls t of the nodes. Nodes are always online. A client is awakened in some round by the environment, requires a number of rounds to synchronize with the rest of the peers and may be killed by the environment at a later round. In each round, the environment

[5]2016, Badr Bellaj, *Is block.timestamp safe for longer time periods?*, Ethereum StackExchange.

invokes EXECUTE on all honest parties and also triggers the adversary. We assume all parties have a *global clock*, meaning they can use the environment function $\mathsf{now}()$ to get the current execution round. We work in two network settings: synchrony and partial synchrony.

Definition 1 (Synchrony). *A network is* synchronous *when all honestly produced messages are delivered with a delay of at most Δ rounds (messages sent during round r arrive by the beginning of round $r + \Delta$).*

Definition 2 (Partial Synchrony). *A network is* partially synchronous *when the adversary can delay messages arbitrarily before a Global Stabilization Time (GST) [10]; however after GST the network becomes synchronous.*

We now turn our attention to distributed ledger protocols.

Definition 3 (Ledger). *A* ledger *is a finite sequence of transactions.*

Definition 4 (Distributed Ledger Protocol). *A* distributed ledger protocol *is an interactive Turing machine which exposes the following methods on each party:*

- EXECUTE$()$*: executes a single round of the protocol, during which the machine can communicate with the network.*
- WRITE(tx)*: takes transaction tx as input.*
- READ$()$*: outputs a ledger.*

The notation $\boldsymbol{L}_r^P$ denotes the output of READ invoked on party P at the end of round r.

Definition 5 (Stickiness). *A distributed ledger protocol execution is* sticky *if for any honest party P and any rounds $r_1 \leq r_2$, it holds that $\boldsymbol{L}_{r_1}^P \preccurlyeq \boldsymbol{L}_{r_2}^P$.*

Definition 6 (Safety). *A distributed ledger protocol execution is* safe *if it is sticky and for any honest parties P_1, P_2 and any rounds r_1, r_2, it holds that $\boldsymbol{L}_{r_1}^{P_1} \sim \boldsymbol{L}_{r_2}^{P_2}$.*

Stickiness can be easily enforced in any safe protocol without stickiness by having the parties report the longest ledger they have seen so far [5].

Definition 7 (Liveness in Synchrony). *A distributed ledger protocol execution, in a synchronous network, is* live(u) *if all transactions written to any honest party at round r, appear in the ledgers of all honest parties by round $r + u$.*

Definition 8 (Liveness in Partial Synchrony). *A distributed ledger protocol execution, in a partially synchronous network, is* live(u) *if all transactions written to any honest party at round r, appear in the ledgers of all honest parties by round* $\max(r, \mathsf{GST}) + u$.

3 Definitions

In this work we are concerned with ledgers that record a round with every transaction indicating the time at which the transaction in question is recorded on the ledger.

Definition 9 (Temporal Ledger). *A* temporal ledger *is a finite sequence of pairs* (r, tx) *where* tx *is a transaction, and* r *is a* round*.*

Definition 10 (Distributed Temporal Ledger Protocol). *A* distributed temporal ledger protocol *is a distributed ledger protocol in which when* READ *is invoked, honest parties output temporal ledgers instead of traditional ledgers.*

The following property of temporal ledgers is central to this work.

Definition 11 (Timely). *A distributed temporal ledger protocol execution is* timely(v) *if for all honest parties* P *and rounds* r_1 *it holds that:*

1. *The rounds recorded in* $\boldsymbol{L}^P_{r_1}$ *are non-decreasing.*
2. *The round recorded at* $\boldsymbol{L}^P_{r_1}[-1]$ *is at most* r_1*.*
3. *For all* $r_1 \le r_2$*, the rounds recorded in* $\boldsymbol{L}^P_{r_2}[|\boldsymbol{L}^P_{r_1}|:]$ *are newer than* $r_1 - v$*.*

Timeliness is orthogonal to safety and liveness. Protocols can have any combination[6] of the three properties.

Definition 12 (Perfectly Timely). *We call a protocol execution* perfectly timely *if it is timely with parameter* $v = 0$*.*

Definition 13 (Supersafety). *A distributed ledger protocol execution is* supersafe *if it is sticky and for any honest parties* P_1, P_2 *and any round* r*, it holds that* $\boldsymbol{L}^{P_1}_r = \boldsymbol{L}^{P_2}_r$*.*

All supersafe protocol executions are safe.

4 Existing Blockchains are Timely

We show that all popularly deployed flavors of blockchain protocols are timely. The vast majority of real blockchain protocols are based on proof-of-work longest chain (Bitcoin, Doge, Litecoin), proof-of-stake longest chain (Cardano), proof-of-stake quorums (Algorand, Cosmos), or a mixture thereof (Ethereum, Polkadot). As exemplary cases we prove that Bitcoin, Ouroboros, and Streamlet are all timely, showing timeliness for each of the respective mechanisms.

[6] A live, timely, but not safe protocol includes all transactions received from the network with the current round. A live, safe, but not timely protocol always records round 1 for all transactions. A safe, timely, but not live protocol always reports empty ledgers.

4.1 From Chains to Ledgers

Blockchain protocols are a type of distributed ledger protocols which includes the three flavors of protocols that we consider. We first define temporal blockchain protocols, a subset of blockchain protocols [19] which keep track of time.

Definition 14 ((Temporal) Blockchain Protocol). *A* blockchain *protocol is a distributed protocol in which, at the end of every round, each honest party outputs a chain. A chain is a finite sequence of blocks. Each block contains a finite sequence of transactions.*

A temporal blockchain *protocol is a blockchain protocol where every block contains a recorded round.*

In blockchain protocols, honest parties at every round output a chain based on their confirmation rule[7] (we also call these *confirmed chains*). We use $\boldsymbol{C}^P_r$ to denote the chain output by party P at the end of round r.

Definition 15 (Chain Safety). *A blockchain protocol execution is* safe *if for all honest parties P_1, P_2 and all rounds r_1, r_2, it holds that $\boldsymbol{C}^{P_1}_{r_1} \sim \boldsymbol{C}^{P_2}_{r_2}$. Furthermore, $\boldsymbol{C}^{P_1}_{r_1} \preccurlyeq \boldsymbol{C}^{P_1}_{r_2}$ (sticky).*

From Temporal Blockchains to Temporal Ledgers. Any temporal blockchain protocol can be transformed into a temporal ledger protocol using the following construction: When READ is invoked on party P at the end of round r, each transaction in $\boldsymbol{C}^P_r$ is reported to $\boldsymbol{L}^P_r$ in the same order as in $\boldsymbol{C}^P_r$, and with the recorded round of the block in which it is included. We call these protocols *chain-based temporal ledger protocols*. Note that an execution of the temporal blockchain protocol corresponds to an execution of the chain-based temporal ledger protocol. Ledger protocols based on safe blockchains are safe.

We now introduce two intermediate properties of temporal blockchains that will help us prove timeliness of chain-based temporal ledger protocols.

Definition 16 (Consistent Recorded Rounds). *A temporal blockchain protocol execution has* consistent recorded rounds *when for all honest parties P and rounds r, the recorded rounds in $\boldsymbol{C}^P_r$ are non-decreasing and not in the future. Additionally, honestly produced blocks*[8] *record the round during which they were produced.*

Definition 17 (Freshness). *A temporal blockchain protocol execution is* fresh*(w) when for any round r, the recorded round r^* of the tip of any honest party's chain output satisfies $r - r^* \leq w$.*

[7]In longest chain protocols, the output is the longest observed chain without the last k blocks. In Streamlet, the output is the longest chain ending in the second of three consecutive notarized blocks with consecutive epoch numbers.

[8]Abstractly, an honestly produced block is a block that was first sent to the network during a broadcast by an honest party. Concretely, in proof-of-work and proof-of-stake, these correspond to honestly mined and honestly proposed blocks respectively. Genesis is considered honestly produced.

The notion of *freshness* tells us that the recorded round of any confirmed chain tip cannot be more than w rounds old. Freshness is a *chain* protocol property. On the other hand, *timeliness* is a *distributed ledger* protocol property, which tells us that newly appearing transactions do not have recorded rounds more than v rounds old. When a ledger protocol is constructed using a chain, these two notions are related through the following theorem.

Theorem 1 (Freshness to Timeliness). *An execution of a temporal ledger protocol based on temporal blockchain protocol whose execution is safe, fresh(w) and has consistent recorded rounds is timely with timeliness $v = w$.*

Proof. Requirements (1) and (2) of timeliness are directly satisfied from the consistent recorded rounds. We now prove (3).

Consider any honest party P, and any rounds $r_1 \leq r_2$. Suppose, towards a contradiction, that $\boldsymbol{L}^P_{r_2}[|\boldsymbol{L}^P_{r_1}|:]$ contains a transaction tx with recorded round $r \leq r_1 - v$. Due to chain safety, it holds that $\boldsymbol{C}^P_{r_1} \prec \boldsymbol{C}^P_{r_2}$. Transaction tx is in some block $B \in \boldsymbol{C}^P_{r_2}[|\boldsymbol{C}^P_{r_1}|:]$ (see Figure 2). Let B^* be the tip of $\boldsymbol{C}^P_{r_1}$ with recorded round r^*. Because of freshness, it holds that $r_1 - r^* \leq w$. Therefore, from $r \leq r_1 - v$ it follows that $r - r^* \leq w - v = 0$.

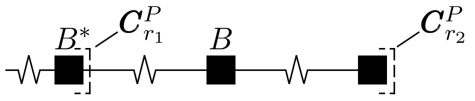

Fig. 2: The chain in the proof of Theorem 1.

This is a contradiction because block B extends a chain that contains B^*, and hence $r > r^*$. □

The following chain property, *recency*, suffices to show freshness[9] and is an intermediate property which will be easier to prove.

Definition 18 (Recency). *A blockchain protocol execution is* recent*(w) when for any round r and honest party P, $\boldsymbol{C}^P_r$ contains a block that was produced by an honest party at most w rounds ago.*

Recency directly yields freshness by the following theorem.

Theorem 2 (Recency to Freshness). *Temporal Blockchains with consistent recorded rounds and recency(w) are fresh(w).*

Proof. Let P be any honest party, r be any round, and r^* be the recorded round of the tip $\boldsymbol{C}^P_r[-1]$. From recency, there is a block $B' \in \boldsymbol{C}^P_r$ honestly produced at round $r' \geq r - w$. Because of consistent recorded rounds, its recorded round is also r'. It follows that $r^* \geq r' \geq r - w$. □

[9] In addition to freshness, recency also suffices to show liveness.

In the next subsections, we express Streamlet, Bitcoin and Ouroboros as chain-based temporal ledger protocols, and we prove that they are recent, thereby showing they are timely.

4.2 Streamlet is Timely

We now prove that the Partially Synchronous Streamlet Protocol is timely, in the partially synchronous setting, after GST. We work in the same model as the Streamlet paper [5], in particular assuming a PKI.

First, we turn Streamlet into a temporal blockchain protocol. Recall Streamlet proceeds in epochs of duration 2Δ rounds. *Temporal Streamlet* is the Streamlet protocol with the following additions: Honest block producers record the first round of the current epoch as the block's recorded round. Blocks with decreasing recorded rounds or recorded rounds in the future are not accepted. Temporal Streamlet remains safe—in the language of [5], *consistent*—and live(u) for some u if $\frac{t}{n} < \frac{1}{3}$. Furthermore, it has consistent recorded rounds.

Honest party P, at the beginning of round r, and before performing any other action, checks the network and outputs the longest confirmed chain $\boldsymbol{C}_r^P$. Let $G(e)$ be the predicate indicating that epochs $e-3, e-2, e-1, e, e+1$ have honest leaders (this predicate is used to prove liveness in [5]). Before proving freshness, we first show the maximum interval between two epochs satisfying $G(e)$ is bounded.

Definition 19 (Temporal Streamlet Typical Execution). *Consider an execution of the Temporal Streamlet protocol with duration of E epochs, n total number of parties and t number of corrupt parties. Let $J = \{0, E\} \cup \{0 < e < E : G(e)\}$. The execution is* typical *if for all $d \geq 5\frac{\kappa + \lg\lfloor\frac{E}{5}\rfloor}{-\lg\left(1-\left(\frac{n-t}{n}\right)^5\right)}$: $\max_{e \in J}(\min_{\substack{e' > e\\ e' \in J}}(e' - e)) \leq d$.*

Lemma 1 (Temporal Streamlet Typicality). *A Temporal Streamlet execution is typical, except with negligible probability in κ.*

Proof. We let $K = \{0, E\} \cup \{e \in \{5, 10, 15, \ldots, 5\lfloor\frac{E}{5}\rfloor\} : G(e)\}$. It holds that $\max_{e\in J}(\min_{\substack{e'>e\\ e'\in J}}(e'-e)) \leq \max_{e\in K}(\min_{\substack{e'>e\\ e'\in K}}(e'-e))$. Therefore:

$$\Pr[\max_{e\in J}(\min_{\substack{e'>e\\ e'\in J}}(e'-e)) > d] \leq$$

$$\Pr[\max_{e\in K}(\min_{\substack{e'>e\\ e'\in K}}(e'-e)) > d] \leq$$

$$\left\lfloor\frac{E}{5}\right\rfloor\left(1-\left(\frac{n-t}{n}\right)^5\right)^{\frac{d}{5}}$$

For the second inequality, we observe that each chunk is a Bernoulli trial and we apply a union bound. Letting $F = \left(1-\left(\frac{n-t}{n}\right)^5\right)$, we obtain $\lfloor\frac{E}{5}\rfloor F^{\frac{d}{5}} \leq 2^{-\kappa} \Rightarrow \frac{d}{5}\lg F \leq -\kappa - \lg\lfloor\frac{E}{5}\rfloor \Rightarrow d \geq 5\frac{\kappa+\lg\lfloor\frac{E}{5}\rfloor}{-\lg F}$. □

Theorem 3 (Streamlet Recency). *A typical execution of Temporal Streamlet is recent with parameter $w = 2\Delta(d+1) - 1$.*

Proof. Let r be the current round of the current epoch e. Let $e' = \max(\{\hat{e} \le e : G(\hat{e})\} \cup \{0\})$. From typicality, it holds that $e - e' \le d$. From the proof of Streamlet liveness [5, Theorem 6], epoch e' contains a confirmed honest block produced at round $r' = 2\Delta(e'-1)+1$. Therefore, $e - e' \le d \Rightarrow r - r' \le 2\Delta(d+1) - 1$. □

Corollary 1 (Streamlet Timeliness). *A typical execution of Temporal Streamlet is timely with parameter $v = 2\Delta(d+1) - 1$.*

Proof. From Theorems 3 and 2, the execution is fresh$(2\Delta(d+1)-1)$. From this, safety [5, Theorem 3], recorded round consistency, and Theorem 1, timeliness with $v = 2\Delta(d+1) - 1$ follows. □

4.3 Longest Chain Protocols are Timely

We now prove that longest chain protocols are timely both in proof-of-work and proof-of-stake. For concreteness, we work in the Bitcoin Backbone [12] setting as a representative of proof-of-work and in the Ouroboros [14] setting as a representative of proof-of-stake (the proof carries over to others in the Ouroboros family [3, 6]). For both, we work with abstract chain virtues and show that any longest chain temporal blockchain protocol with three crucial properties—Common Prefix, Chain Quality, and Chain Growth [12]—is timely, provided it has consistent recorded rounds. In this subsection, we are in the static difficulty/stake and synchronous setting with $\Delta = 1$.

Definition 20 (Longest Chain Protocol). *A longest chain protocol with confirmation parameter k is a blockchain protocol for which, at the beginning of any round r, every honest party P* adopts[10] *the* longest *valid observed unstable chain $\tilde{\boldsymbol{C}}_r^P$. It outputs the confirmed chain $\boldsymbol{C}_r^P = \tilde{\boldsymbol{C}}_r^P[:-k]$. Whenever an honest party generates a block, it extends its adopted unstable chain. This new chain is broadcast and guaranteed to be valid.*

For the notation definitions (μ, ℓ, τ, s) in the following lemma, refer to Figure 3 and the original Bitcoin Backbone paper [11].

Lemma 2 (Longest Chain Recency). *Blockchain protocols following the longest chain rule, with* Chain Quality(μ, ℓ) *and* Chain Growth(τ, s) *are recent with parameter $w = \max(s, \frac{k+\ell}{\tau}) + 1$.*

Proof. Let P be any honest party and r be any round. Let B' be the most recent honestly generated block in $\boldsymbol{C}_r^P[-\ell:]$ (or let B' be genesis if $|\boldsymbol{C}_r^P[-\ell:]| \le \ell$). This

[10] At the beginning of a round r, before observing the network, we say that honest party P *has* an unstable chain $\hat{\boldsymbol{C}}_r^P$ (this chain contains any block honestly generated by P at the previous round), whereas upon observing the network, the honest party *adopts* the longest valid unstable chain $\tilde{\boldsymbol{C}}_r^P$.

μ: Chain Quality parameter (ratio of honest-to-total blocks in a chain).
ℓ: Number of blocks in a chunk for Chain Quality to apply.
τ: Chain Growth parameter (how much the chain grows, in blocks per round).
s: Number of rounds for Chain Growth to apply.

Fig. 3: Overview of Bitcoin Backbone / Ouroboros variables μ, ℓ, τ, s.

block exists by Chain Quality because we are looking at a chain chunk of length at least ℓ and $\mu\ell \geq 1$ (or B' is genesis). Let r' be the round in which B' was generated, and P' be the party who generated it (or $P' = P, r' = 0$ if B' is genesis). Suppose, towards a contradiction, that

$$r' < r - w\,. \tag{1}$$

Let $\tilde{\boldsymbol{C}}^{P'}_{r'}$ be the chain that P' adopts at round r' (this will be the empty chain if B' is genesis). Party P' extends $\tilde{\boldsymbol{C}}^{P'}_{r'}$, at round r', with block B', creating a chain of length $|\,\tilde{\boldsymbol{C}}^{P}_{r'}|+1$. This newly generated chain is broadcast to the network and received by party P at the beginning of round $r'+1$. Let $\hat{\boldsymbol{C}}^{P}_{r'+2}$ be the chain that P *has* at round $r'+2$. We observe that, at round $r'+2$, due to the longest chain rule, party P *has* a chain of greater or equal length to the one broadcast by party P'. Hence, $|\,\hat{\boldsymbol{C}}^{P}_{r'+2}| \geq |\,\tilde{\boldsymbol{C}}^{P'}_{r'}| + 1$. Therefore

$$|\,\tilde{\boldsymbol{C}}^{P}_{r}| - |\,\hat{\boldsymbol{C}}^{P}_{r'+2}| \leq |\,\tilde{\boldsymbol{C}}^{P}_{r}| - |\,\tilde{\boldsymbol{C}}^{P'}_{r'}| - 1 < k + \ell\,. \tag{2}$$

For the second inequality, refer to Figure 4.

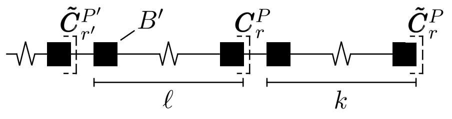
$\tilde{\boldsymbol{C}}^{P'}_{r'}$
B'
$\boldsymbol{C}^{P}_{r}$
$\tilde{\boldsymbol{C}}^{P}_{r}$
ℓ
k

Fig. 4: Longest chain recency proof. Block B' is illustrated in the earliest possible position.

On the other hand, by Inequality 1, $r - (r' + 2) \geq w - 1 \geq s$ and we can apply Chain Growth between rounds $r'+2$ and r with parameters s, τ to obtain $|\,\tilde{\boldsymbol{C}}^{P}_{r}| - |\,\hat{\boldsymbol{C}}^{P}_{r'+2}| \geq \tau(r - (r' + 2)) \geq \tau(w - 1) \geq k + \ell$, which is a contradiction because of Inequality (2). $\square$

Ouroboros is already a Temporal Blockchain protocol with consistent recorded rounds. The Bitcoin Backbone protocol can be augmented in a straightforward manner to include recorded rounds, as done in the real Bitcoin deployment [2]. The augmentation is illustrated in Figure 5. The construction retains Common Prefix, Chain Quality and Chain Growth. Furthermore, it has consistent recorded rounds.

The Temporal Bitcoin protocol is the Bitcoin protocol with the following additions:

1. Blocks include a round number. Genesis has recorded round 0.
2. Honest parties mine blocks with the current round.
3. The recorded rounds of a valid chain are strictly increasing and not in the future.

Fig. 5: Temporal Bitcoin construction.

Corollary 2 (Bitcoin and Ouroboros Timeliness). *A typical execution of Temporal Bitcoin or Ouroboros is timely with parameter* $v = \max(s, \frac{k+\ell}{\tau}) + 1$.

Proof. Safety follows from Chain safety, which follows from Common Prefix [12, Theorem 15] [14, Theorem 4.31]. From Chain Quality(μ, ℓ) [12, Theorem 16] [14, Lemma 4.19], Chain Growth(τ, s) [12, Theorem 12] [14, Lemma 4.22], and Theorem 2, the execution is recent($\max(s, \frac{k+\ell}{\tau})+1$). From recorded round consistency and Theorem 2, it is fresh($\max(s, \frac{k+\ell}{\tau}) + 1$). From this, safety, recorded round consistency and Theorem 1, timeliness($\max(s, \frac{k+\ell}{\tau}) + 1$) follows. □

5 Impossibility

We saw that in the synchronous setting and in the partially synchronous setting after GST, protocols can be timely. Conversely, in the following theorem, we show that timeliness before GST in partially synchronous networks is unachievable.

Theorem 4 (Timeliness is Impossible in Partial Synchrony). *In partial synchrony, a safe and live*(u) *distributed temporal ledger protocol cannot be timely before GST.*

Proof. Suppose, towards a contradiction, there is a safe, live(u), and timely(v) distributed temporal ledger protocol with n parties. Consider the following two worlds:
World A. There are $n-1$ honest parties and one adversarial party. The adversary remains silent and does not disrupt the synchrony of the network ($\mathsf{GST} = 0$). The adversary chooses any round r and introduces a high entropy transaction tx to the honest parties. We are in synchrony, hence, because of liveness, tx appears in the ledgers of all honest parties by round $r + u$. The transaction is recorded with round $r^* \leq r + u$ because of timeliness.
World B. All parties are honest. We are under partial synchrony, and the adversary partitions the network such that one honest party P is isolated from the other $n - 1$ honest parties. The adversary sets GST to $r + u + v + 1$. Before GST, at round r, the transaction tx is introduced to the $n - 1$ connected honest parties. In the view of these parties, World A and World B are identical up to

now. Hence, like in World A, they report tx in their ledgers by round $r+u$, with recorded round $r^* \le r+u$.

At $r_2 \ge \mathsf{GST}$, due to liveness, tx will be included in P's ledger for the first time, with recorded round r^* due to safety. Hence, $(r^*, \mathsf{tx}) \in \boldsymbol{L}^P_{r_2}[|\boldsymbol{L}^P_{\mathsf{GST}-1}|:]$. From timeliness, we get $r^* > \mathsf{GST} - v - 1$. Therefore, $r^* > r+u$. This is a contradiction, since $r^* \le r+u$. □

6 Timeliness to Supersafety and Back

6.1 Timeliness → Supersafety

We now construct a supersafe protocol Π^* using a black-box reduction from a safe, live(u) and timely(v) protocol Π. Each honest party P, executing the Π^* protocol, runs a full node of protocol Π. The ledger of party P for protocol Π and Π^* is denoted as $\boldsymbol{L}^P_r$ and ${}^*\boldsymbol{L}^P_r$ respectively. The main idea is that ledger ${}^*\boldsymbol{L}_r$ is constructed by filtering through ledger $\boldsymbol{L}_r$, and only keeping transactions with recorded round less than or equal to $r-v$. The reduction is illustrated in Figure 6 and Algorithm 1.

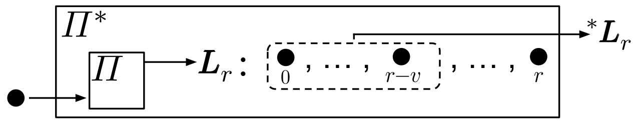

Fig. 6: The reduction from Timeliness (the Π protocol) to Supersafety (the Π^* protocol). A transaction is illustrated as a black circle and its recorded round is displayed below.

When the READ function is invoked, the temporal ledger of party P is acquired in Line 6. Then, transactions with recorded round less than or equal to $\mathsf{now}() - v$ are filtered to create ${}^*\boldsymbol{L}$, which is returned in Line 7. This introduces a confirmation delay of v additional rounds (see Theorem 5). When the WRITE function is invoked with transaction tx, party P^* simply writes tx to party P in Line 10.

Algorithm 1 The reduction from Timeliness (the Π protocol) to Supersafety (the Π^* protocol).

```
1: $P \leftarrow$ **new** $\Pi()$
2: **function** EXECUTE()
3:     $P$.EXECUTE()
4: **end function**
5: **function** READ()
6:     $\boldsymbol{L} \leftarrow P$.READ()
7:     **return** $[(r, \mathsf{tx}) \in \boldsymbol{L} : r \leq \mathsf{now}() - v]$
8: **end function**
9: **function** WRITE($\mathsf{tx}$)
10:     $P$.WRITE($\mathsf{tx}$)
11: **end function**
```

Before proving protocol Π^* is supersafe, we present the following lemma[11], which allows us to argue that all parties in protocol Π share a common view of sufficiently old transactions.

Lemma 3 (Past Perfect). *Consider a safe, live(u), and timely(v) temporal ledger protocol execution with duration R rounds. If for some honest party P_1 and some round r_1 it holds that $(r^*, \mathsf{tx}) \in \boldsymbol{L}_{r_1}^{P_1}$, then for all honest parties P_2 and for all rounds $r_2 \geq r^* + v$ it holds that $(r^*, \mathsf{tx}) \in \boldsymbol{L}_{r_2}^{P_2}$, as long as at least one new honest transaction tx' appears in the network at any round r_3, where $r_1 < r_3 \leq R - u$.*

Proof. Consider an execution as in the statement and suppose, towards a contradiction, that $(r^*, \mathsf{tx}) = \boldsymbol{L}_{r_1}^{P_1}[k]$ for some $k \in \mathbb{N}$, but $(r^*, \mathsf{tx}) \notin \boldsymbol{L}_{r_2}^{P_2}$ with $r_2 \geq r^* + v$. From safety, $\boldsymbol{L}_{r_2}^{P_2} \prec \boldsymbol{L}_{r_1}^{P_1}$ and $|\boldsymbol{L}_{r_2}^{P_2}| \leq k < |\boldsymbol{L}_{r_1}^{P_1}|$. Due to liveness, $(r', \mathsf{tx}') = \boldsymbol{L}_{r_3+u}^{P_2}[k']$, for some $r', k' \in \mathbb{N}$. As tx' is new, it is not in $\boldsymbol{L}_{r_1}^{P_1}$. Due to safety, $k' \geq |\boldsymbol{L}_{r_1}^{P_1}| > k$, and $\boldsymbol{L}_{r_3+u}^{P_2}[k] = (r^*, \mathsf{tx})$. Therefore, $(r^*, \mathsf{tx}) \in \boldsymbol{L}_{r_3+u}^{P_2}[|\boldsymbol{L}_{r_2}^{P_2}|:]$. From liveness, it must hold that $r_2 < r_3 + u$. Since $r^* \leq r_2 - v$, this contradicts the timeliness with parameter v. ◻

Theorem 5 (Timeliness to Supersafety). *An execution of Π^*, with duration R rounds, is supersafe and live($u + v$), if the execution of Π is safe, live(u) and timely(v) as long as at least one new honest transaction appears in the network at each round before $R - u$.*

Proof. Consider any honest parties P_1, P_2, any round r, and any $(r^*, \mathsf{tx}) \in \boldsymbol{L}_r^{P_1}$, where $r^* \leq r - v$. From Lemma 3 it holds that $(r^*, \mathsf{tx}) \in \boldsymbol{L}_r^{P_2}$. From this and from safety, it follows that ${}^*\boldsymbol{L}_r^{P_1} = [(r^*, \mathsf{tx}) \in \boldsymbol{L}_r^{P_1} : r^* \leq r - v] \preccurlyeq [(r^*, \mathsf{tx}) \in \boldsymbol{L}_r^{P_2} : r^* \leq r - v] = {}^*\boldsymbol{L}_r^{P_2}$. Inversely, ${}^*\boldsymbol{L}_r^{P_2} \preccurlyeq {}^*\boldsymbol{L}_r^{P_1}$. Therefore, ${}^*\boldsymbol{L}_r^{P_1} = {}^*\boldsymbol{L}_r^{P_2}$. Lastly, stickiness follows from the stickiness of Π. Therefore, Π^* is supersafe. Liveness follows from the liveness of Π with an added confirmation delay of v rounds. ◻

[11] A variant of this auxiliary lemma was introduced and proven in earlier work [22].

Note that a new transaction in every round is required for the purpose of the reduction and is not required for building a supersafe protocol in practice. We note that the above theorem also applies to late-joining clients.

6.2 Supersafety → Perfect Timeliness

We now construct a perfectly timely protocol Π^* using a black-box reduction from a supersafe, and live(u) protocol Π. Each honest party P, executing the Π^* protocol, runs a full node of protocol Π. The main idea is that, even though Π is not a temporal ledger, since it is supersafe, we can simply ascribe to each new transaction the round at which it first appeared on our ledger without loss of safety. This reduction is illustrated in Figure 7 and Algorithm 2.

Algorithm 2 The reduction from Supersafety (the Π protocol) to Perfect Timeliness (the Π^* protocol).

```
1: $P \leftarrow$ new $\Pi()$
2: $^*\boldsymbol{L} \leftarrow [\,]$
3: function EXECUTE()
4:     $P$.EXECUTE()
5:     $\boldsymbol{L} \leftarrow P$.READ()
6:     for $\mathsf{tx} \in \boldsymbol{L}[|^*\boldsymbol{L}|:]$ do
7:         $^*\boldsymbol{L} \leftarrow {}^*\boldsymbol{L} \,\|\, (\mathsf{now}(), \mathsf{tx})$
8:     end for
9: end function
10: function READ()
11:     return $^*\boldsymbol{L}$
12: end function
13: function WRITE(tx)
14:     $P$.WRITE(tx)
15: end function
```

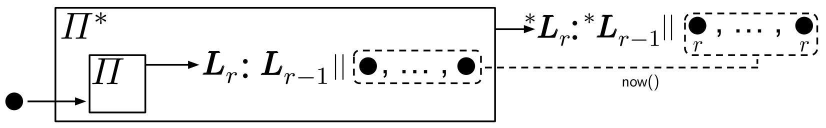
Π^*
Π
$\boldsymbol{L}_r\!: \boldsymbol{L}_{r-1} \| \bullet, \dots, \bullet$
$^*\boldsymbol{L}_r\!: {}^*\boldsymbol{L}_{r-1} \| \bullet, \dots, \bullet$
r
r
now()

Fig. 7: The reduction from Supersafety (the Π protocol) to Perfect Timeliness (the Π^* protocol). New transactions of $\boldsymbol{L}_r$ are included in $^*\boldsymbol{L}_r$ with recorded round r.

Theorem 6 (Supersafety to Perfect Timeliness). *An execution of Π^* is perfectly timely, supersafe, and live(u), if the execution of Π is supersafe and live(u).*

Proof. Timeliness requirements (1) and (2) are directly satisfied. For (3) consider any honest party P and any rounds $r_1 \leq r_2$. Only transactions with recorded round greater than r_1 appear in ledger ${}^*\boldsymbol{L}^P_{r_2}[|{}^*\boldsymbol{L}^P_{r_1}|:]$. Hence, Π^* is timely with parameter $v = 0$. Supersafety and liveness(u) follow from those of Π. □

6.3 A Perfectly Timely, Supersafe Protocol with Clients

The protocol of the last subsection is perfectly time and supersafe. But how about late-joining clients? When a client first boots, it obtains all past transactions from Π. However, since Π is not temporal, the client does not see any recorded rounds associated with the transactions. Additionally, since the recorded rounds depend on past ledger append times, the client cannot deduce them on its own.

To enable client support, we make a small addition to the protocol: Each node runs an *additional* instance of the internal ledger protocol Π to facilitate the booting of clients. We call Π_1 the first instance of Π (which operates as before), and Π_2 this new instance of Π. The output ledger ${}^*\boldsymbol{L}$ of the nodes remains the same. Since the online nodes have observed the rounds with which transactions of Π_1 are reported, they record those rounds on the second ledger Π_2 to help the client recover them. Let ${}^1\boldsymbol{L}$ and ${}^2\boldsymbol{L}$ be the output ledgers of Π_1 and Π_2 respectively. The instance Π_2 is used as follows: In every round r, whenever new transactions appear in ${}^*\boldsymbol{L}_r$, honest nodes introduce one transaction tx' in Π_2 for each newly appearing transaction tx of ${}^*\boldsymbol{L}_r$. The payload of tx' contains the pair $\langle \mathsf{tx}, r \rangle$ (and note that, here, r is part of the transaction tx' payload, as Π_2 is not a temporal ledger). We parameterize Π_2 with the following transaction validity language: Valid transactions tx' must take the form $\langle \mathsf{tx}, r \rangle$, where tx first appeared in ${}^*\boldsymbol{L}$ at round r. Otherwise, the transaction is rejected as invalid. The protocol is illustrated in Figure 8.

A late-joining client first synchronizes Π_1 and Π_2 with the rest of the peers. Due to safety, liveness and the validity rule of Π_2, the client can read ledger ${}^2\boldsymbol{L}$ to find out the recorded rounds of past transactions. Eventually, all past transaction recorded rounds are included in ${}^2\boldsymbol{L}$ and the client can construct the correct temporal ledger ${}^*\boldsymbol{L}$. At this point we say the client is fully booted and can continue operating as a full online node without Π_2, i.e., it outputs the ledger ${}^*\boldsymbol{L}$. From this point on, the client is perfectly timely and supersafe.

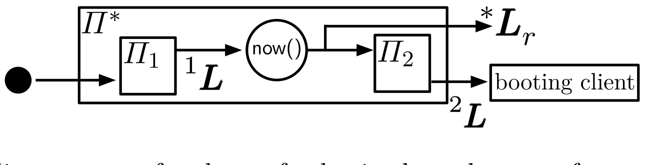

Fig. 8: Client support for the perfectly timely and supersafe protocol $\Pi*$.

Lastly, to make this protocol efficient, we observe that the payload of Π_2 transactions does not need to include the whole Π_1 transaction, but it is sufficient to include the transaction id only. Lastly, the two instances of Π can be combined into one by coloring transactions appropriately: A normal transaction tx, colored *black*, enters into the Π^* protocol and is recorded on Π as usual. When it stabilizes and its round has been recorded by an online node, this node wraps the transaction tx into an *orange* transaction tx' containing as payload tx together with its recorded round. The ledger output by Π is split into black and orange parts. The black part is reported as the output ledger of Π^*. The orange part is used to help clients boot, and is ignored by a client after booting has finished. The validity language restrictions only apply to the orange transactions. This final construction is illustrated in Figure 9.

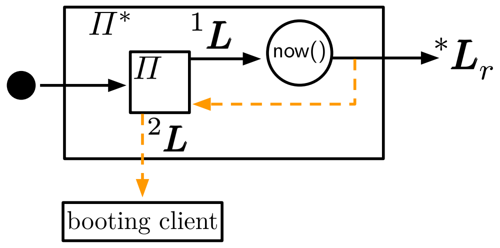

Fig. 9: The client-supporting supersafe and perfectly timely protocol Π^* using only one instance of Π for efficiency. Normal transactions and ledgers are shown in solid black lines; auxiliary transactions are shown in dashed orange lines.

7 Conclusion

We introduced two new properties of ledger protocols: Timeliness and supersafety. Timeliness mandates that transactions are recorded on ledgers with a timestamp that roughly corresponds to the current real time. Supersafety mandates that all parties report ledgers that are identical to each other at every moment. We proved that all popular blockchain deployments attain timeliness under the assumption that clocks are synchronized, and the network is synchronous. We also showed that the synchrony assumption is a necessary condition. We reduced supersafety to timeliness and vice versa. Our final protocol is live, supersafe, perfectly timely, and has support for late-joining clients. Applying both our reductions in series to *any* blockchain protocol with safety, liveness, timeliness and client support allows transforming it into a protocol that additionally enjoys supersafety and perfect timeliness. We hope that the nomenclature developed in this paper will shed some light on the constituent parts of security proofs of previous protocols and simplify the design of future ones.

8 Acknowledgements

We thank Giulia Scaffino for reading early versions of this paper and providing detailed feedback; Ertem Nusret Tas for providing context on the relevance of timeliness in Babylon and early discussions on timeliness; David Tse for early discussions on timeliness and supersafety; and Orfeas Stefanos Thyfronitis Litos for reading early versions of this paper and providing feedback, as well as insights on the relevance of timeliness and supersafety in the context of payment channels. This research was partially funded by a Research Hub Collaboration agreement with Input Output Global Inc.

References

1. H. K. Alper. Network time with a consensus on clock. *Cryptology ePrint Archive*, 2019.
2. A. M. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media, 2014.
3. C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas. Ouroboros Genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930, 2018.
4. C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. Ouroboros Chronos: Permissionless Clock Synchronization via Proof-of-Stake. *Cryptology ePrint Archive*, 2019.
5. B. Y. Chan and E. Shi. Streamlet: Textbook Streamlined Blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 1–11, 2020.
6. B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.
7. B. David, B. Magri, C. Matt, J. B. Nielsen, and D. Tschudi. GearBox: Optimal-size Shard Committees by Leveraging the Safety-Liveness Dichotomy. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 683–696. ACM, 2022.
8. E. Dimitrova. 0x: ZRXWrappedToken. Available at: `https://github.com/0xProject/protocol/blob/b19e29e03d8a6bf5d797af18c3ce227594994f55/contracts/governance/src/ZRXWrappedToken.sol#L130`, Mar 2023.
9. D. Dolev and H. R. Strong. Authenticated Algorithms for Byzantine Agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
10. C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the Presence of Partial Synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.
11. J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications (revised 2019). Cryptology ePrint Archive, Report 2014/765, 2014. `https://eprint.iacr.org/2014/765`.
12. J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In E. Oswald and M. Fischlin, editors, *Annual International*

Conference on the Theory and Applications of Cryptographic Techniques, volume 9057 of *LNCS*, pages 281–310. Springer, Apr 2015.

13. J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In J. Katz and H. Shacham, editors, *Annual International Cryptology Conference*, volume 10401 of *LNCS*, pages 291–323. Springer, Aug 2017.
14. A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In J. Katz and H. Shacham, editors, *Annual International Cryptology Conference*, volume 10401 of *LNCS*, pages 357–388. Springer, Springer, Aug 2017.
15. L. Lamport and P. M. Melliar-Smith. Synchronizing Clocks in the Presence of Faults. *Journal of the ACM (JACM)*, 32(1):52–78, 1985.
16. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.
17. S. Nakamoto. Bitcoin source code: chain.h. Available at: `https://github.com/bitcoin/bitcoin/blob/8247a8db6963d2116dc4697a3217d736c197f91d/src/chain.h#L24`, 2009.
18. E. Nusret Tas, D. Tse, F. Yu, and S. Kannan. Babylon: Reusing Bitcoin Mining to Enhance Proof-of-Stake Security. *arXiv e-prints*, pages arXiv–2201, 2022.
19. R. Pass and E. Shi. Rethinking Large-Scale Consensus. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 115–129. IEEE, 2017.
20. P. Szalachowski. (Short Paper) Towards More Reliable Bitcoin Timestamps. In *2018 crypto valley conference on blockchain technology (CVCBT)*, pages 101–104. IEEE, 2018.
21. J. Wilcke. Geth source code: blockchain.go. Available at: `https://github.com/ethereum/go-ethereum/blob/5b9cbe30f8ca2487c8991e50e9c939d5e6ec3cc2/core/blockchain.go#L1557`, 2015.
22. D. Zindros, A. Tzinas, and D. Tse. ROLLERBLADE: Replicated Distributed Protocol Emulation on Top of Ledgers. Unpublished Manuscript, Aug. 2023.