Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing

Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford, *École Polytechnique Fédérale de Lausanne (EPFL)*

https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias

This paper is included in the Proceedings of the 25th USENIX Security Symposium

August 10–12, 2016 • Austin, TX

ISBN 978-1-931971-32-4

Open access to the Proceedings of the 25th USENIX Security Symposium is sponsored by USENIX

Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing

Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly,
Ismail Khoffi, Linus Gasser, and Bryan Ford
EPFL

Abstract

While showing great promise, Bitcoin requires users to wait tens of minutes for transactions to commit, and even then, offering only probabilistic guarantees. This paper introduces ByzCoin, a novel Byzantine consensus protocol that leverages scalable collective signing to commit Bitcoin transactions irreversibly within seconds. ByzCoin achieves Byzantine consensus while preserving Bitcoin's open membership by dynamically forming hash power-proportionate consensus groups that represent recently-successful block miners. ByzCoin employs communication trees to optimize transaction commitment and verification under normal operation while guaranteeing safety and liveness under Byzantine faults, up to a near-optimal tolerance of f faulty group members among $3f+2$ total. ByzCoin mitigates double spending and selfish mining attacks by producing collectively signed transaction blocks within one minute of transaction submission. Tree-structured communication further reduces this latency to less than 30 seconds. Due to these optimizations, ByzCoin achieves a throughput higher than Paypal currently handles, with a confirmation latency of 15-20 seconds.

1 Introduction

Bitcoin [47] is a decentralized cryptocurrency providing an open, self-regulating alternative to classic currencies managed by central authorities such as banks. Bitcoin builds on a peer-to-peer network where users can submit transactions without intermediaries. Special nodes, called *miners*, collect transactions, solve computational puzzles (*proof-of-work*) to reach consensus, and add the transactions in form of blocks to a distributed public ledger known as the *blockchain*.

The original Bitcoin paper argues that transaction processing is secure and irreversible, as long as the largest colluding group of miners represents less than 50% of total computing capacity and at least about one hour has elapsed. This high transaction-confirmation latency limits Bitcoin's suitability for real-time transactions. Later work revealed additional vulnerabilities to transaction reversibility, double-spending, and strategic mining attacks [25, 31, 34, 35, 48, 3].

The key problem is that Bitcoin's consensus algorithm provides only probabilistic consistency guarantees. Strong consistency could offer cryptocurrencies three important benefits. First, all miners instantly agree on the validity of blocks, without wasting computational power resolving inconsistencies (*forks*). Second, clients need not wait for extended periods to be certain that a submitted transaction is committed; as soon as it appears in the blockchain, the transaction can be considered confirmed. Third, strong consistency provides *forward security*: as soon as a block has been appended to the blockchain, it stays there forever. Although increasing the consistency of cryptocurrencies has been suggested before [17, 19, 43, 52, 56], existing proposals give up Bitcoin's decentralization, and/or introduce new and non-intuitive security assumptions, and/or lack experimental evidence of performance and scalability.

This work introduces ByzCoin, a Bitcoin-like cryptocurrency enhanced with strong consistency, based on the principles of the well-studied Practical Byzantine Fault Tolerance (PBFT) [14] algorithm. ByzCoin addresses four key challenges in bringing PBFT's strong consistency to cryptocurrencies: (1) open membership, (2) scalability to hundreds of replicas, (3) proof-of-work block conflicts, and (4) transaction commitment rate.

PBFT was not designed for scalability to large consensus groups: deployments and experiments often employ the minimum of four replicas [38], and generally have not explored scalability levels beyond 7 [14] or 16 replicas [16, 32, 1]. ByzCoin builds PBFT atop CoSi [54], a collective signing protocol that efficiently aggregates hundreds or thousands of signatures. Collective signing reduces both the costs of PBFT rounds and the costs

for "light" clients to verify transaction commitment. Although CoSi is not a consensus protocol, ByzCoin implements Byzantine consensus using CoSi signing rounds to make PBFT's *prepare* and *commit* phases scalable.

PBFT normally assumes a well-defined, closed group of replicas, conflicting with Bitcoin's open membership and use of proof-of-work to resist Sybil attacks [23]. ByzCoin addresses this conflict by forming consensus groups dynamically from *windows* of recently mined blocks, giving recent miners *shares* or voting power proportional to their recent commitment of hash power. Lastly, to reduce transaction processing latency we adopt the idea from Bitcoin-NG [24] to decouple transaction verification from block mining.

Experiments with a prototype implementation of ByzCoin show that a consensus group formed from approximately the past 24 hours of successful miners (144 miners) can reach consensus in less than 20 seconds, on blocks of Bitcoin's current maximum size (1MB). A larger consensus group formed from one week of successful miners (1008) reached consensus on an 8MB block in 90 seconds, showing that the systems scales both with the number of participants and with the block size. For the 144-participant consensus group, with a block size of 32MB, the system handles 974 transactions per second (TPS) with a 68-second confirmation latency. These experiments suggest that ByzCoin can handle loads higher than PayPal and comparable with Visa.

ByzCoin is still a proof-of-concept with several limitations. First, ByzCoin does not improve on Bitcoin's proof-of-work mechanism; finding a suitable replacement [4, 28, 37, 58] is an important but orthogonal area for future work. Like many BFT protocols in practice [15, 32], ByzCoin is vulnerable to slowdown or temporary DoS attacks that Byzantine nodes can trigger. Although a malicious leader cannot violate or permanently block consensus, he might temporarily exclude minority sets ($< \frac{1}{3}$) of victims from the consensus process, depriving them of rewards, and/or attempt to censor transactions. ByzCoin guarantees security only against attackers who consistently control less than a third (not 50%) of consensus group shares – though Bitcoin has analogous weaknesses accounting for selfish mining [25].

In this paper we make the following key contributions:

- We use collective signing [54] to scale BFT protocols to large consensus groups and enable clients to verify operation commitments efficiently.
- We present (§3) the first demonstrably practical Byzantine consensus protocol supporting not only static consensus groups but also dynamic membership proportional to proof-of-work as in Bitcoin.
- We demonstrate experimentally (§4) that a strongly-consistent cryptocurrency can increase Bitcoin's throughput by two orders of magnitude, with a transaction confirmation latency under one minute.
- We find through security analysis (§5) that ByzCoin can mitigate several known attacks on Bitcoin provided no attacker controls more than $\frac{1}{4}$ of hash power.

2 Background and Motivation

This section first outlines the three most relevant areas of prior work that ByzCoin builds on: cryptocurrencies such as Bitcoin and Bitcoin-NG, Byzantine fault tolerance (BFT) principles, and collective signing techniques.

2.1 Bitcoin and Variations

Bitcoin. At the core of Bitcoin [47] rests the so-called *blockchain*, a public, append-only database maintained by *miners* and serving as a global ledger of all transactions ever issued. Transactions are bundled into *blocks* and validated by a *proof-of-work*. A block is valid if its cryptographic hash has d leading zero bits, where the difficulty parameter d is adjusted periodically such that new blocks are mined about every ten minutes on average. Each block includes a Merkle tree [44] of new transactions to be committed, and a cryptographic hash chaining to the last valid block, thereby forming the blockchain. Upon successfully forming a new block with a valid proof-of-work, a miner broadcasts the new block to the rest of the miners, who (when behaving properly) accept the new block, if it extends a valid chain strictly longer than any they have already seen.

Bitcoin's decentralized consensus and security derive from an assumption that a majority of the miners, measured in terms of *hash power* or ability to solve hash-based proof-of-work puzzles, follows these rules and always attempts to extend the longest existing chain. As soon as a quorum of miners with the majority of the network's hash power approves a given block by mining on top of it, the block remains embedded in any future chain [29]. Bitcoin's security is guaranteed by the fact that this majority will be extending the legitimate chain faster than any corrupt minority that might try to rewrite history or double-spend currency. However, Bitcoin's consistency guarantee is only probabilistic, which leads to two fundamental problems.

First, multiple miners might find distinct blocks with the same parent before the network has reached consensus. Such a conflict is called a *fork*, an inconsistency that is temporarily allowed until one of the chains is extended yet again. Subsequently, all well-behaved miners on the shorter chain(s) switch to the new longest one. All transactions appearing only in the rejected block(s) are invalid and must be resubmitted for inclusion into the winning blockchain. This means that Bitcoin clients who want high certainty that a transaction is complete (*e.g.*, that

they have irrevocably received a payment) must wait not only for the next block but for several blocks thereafter, thus increasing the time interval until a transaction can be considered complete. As a rule of thumb [47], a block is considered as permanently added to the blockchain after about 6 new blocks have been mined on top of it, for a confirmation latency of 60 minutes on average.

Second, the Bitcoin block size is currently limited to 1 MB. This limitation in turn results in an upper bound on the number of transactions per second (TPS) the Bitcoin network can handle, estimated to be an average of 7 TPS. For comparison, Paypal handles 500 TPS and VISA even 4000 TPS. An obvious solution to enlarge Bitcoin's throughput is to increase the size of its blocks. Unfortunately, this solution also increases the probability of forks due to higher propagation delays and the risk of double-spending attacks [53, 30, 36]. Bitcoin's liveness and security properties depend on forks being relatively rare. Otherwise, the miners would spend much of their effort trying to resolve multiple forks [31, 17], or in the extreme case, completely centralize Bitcoin [24]

Bitcoin-NG. Bitcoin-NG [24] makes the important observation that Bitcoin blocks serve two different purposes: (1) election of a leader who decides how to resolve potential inconsistencies, and (2) verification of transactions. Due to this observation, Bitcoin-NG proposes two different block types: *Keyblocks* are generated through mining with proof-of-work and are used to securely elect leaders, at a moderate frequency, such as every 10 minutes as in Bitcoin. *Microblocks* contain transactions, require no proof-of-work, and are generated and signed by the elected leader. This separation enables Bitcoin-NG to process many microblocks between the mining of two keyblocks, enabling transaction throughput to increase.

Bitcoin-NG, however, retains many drawbacks of Bitcoin's consistency model. Temporary forks due to near-simultaneous keyblock mining, or deliberately introduced by selfish or malicious miners, can still throw the system into an inconsistent state for 10 minutes or more. Further, within any 10-minute window the current leader could still intentionally fork or rewrite history and invalidate transactions. If a client does not wait several tens of minutes (as in Bitcoin) for transaction confirmation, he is vulnerable to double-spend attacks by the current leader or by another miner who forks the blockchain. Although Bitcoin-NG includes disincentives for such behavior, these disincentives amount at most to the "mining value" of the keyblock (coinbase rewards and transaction fees): Thus, leaders are both able and have incentives to double-spend on higher-value transactions.

Consequently, although Bitcoin-NG permits higher transaction throughput, it does not solve Bitcoin's consistency weaknesses. Nevertheless, Bitcoin-NG's decoupling of keyblocks from microblocks is an important idea that we build on in Section 3.6 to support high-throughput and low-latency transactions in ByzCoin.

2.2 Byzantine Fault Tolerance

The *Byzantine Generals' Problem* [39, 49] refers to the situation where the malfunctioning of one or several components of a distributed system prevents the latter from reaching an agreement. Pease et al. [49] show that $3f+1$ participants are necessary to be able to tolerate f faults and still reach consensus. The *Practical Byzantine Fault Tolerance (PBFT)* algorithm [14] was the first efficient solution to the Byzantine Generals' Problem that works in weakly synchronous environments such as the Internet. PBFT offers both *safety* and *liveness* provided that the above bound applies, *i.e.*, that at most f faults among $3f+1$ participants occur. PBFT triggered a surge of research on Byzantine replication algorithms with various optimizations and trade-offs [1, 16, 38, 32].

Every round of PBFT has three distinct phases. In the first, *pre-prepare* phase, the current primary node or *leader* announces the next record that the system should agree upon. On receiving this pre-prepare, every node validates the correctness of the proposal and multicasts a *prepare* message to the group. The nodes wait until they collect a quorum of $(2f+1)$ prepare messages and publish this observation with a *commit* message. Finally, they wait for a quorum of $(2f+1)$ commit messages to make sure that enough nodes have recorded the decision.

PBFT relies upon a correct leader to begin each round and proceeds if a two-thirds quorum exists; consequently, the leader is an attack target. For this reason PBFT has a view-change protocol that ensures liveness in the face of a faulty leader. All nodes monitor the leader's actions and if they detect either malicious behavior or a lack of progress, initiate a view-change. Each node independently announces its desire to change leaders and stops validating the leader's actions. If a quorum of $(2f+1)$ nodes decides that the leader is faulty, then the next leader in a well-known schedule takes over.

PBFT has its limitations. First, it assumes a fixed, well-defined group of replicas, thus contradicting Bitcoin's basic principle of being decentralized and open for anyone to participate. Second, each PBFT replica normally communicates directly with every other replica during each consensus round, resulting in $O(n^2)$ communication complexity: This is acceptable when n is typically 4 or not much more, but becomes impractical if n represents hundreds or thousands of Bitcoin nodes. Third, after submitting a transaction to a PBFT service, a client must communicate with a super-majority of the replicas in order to confirm the transaction has been com-

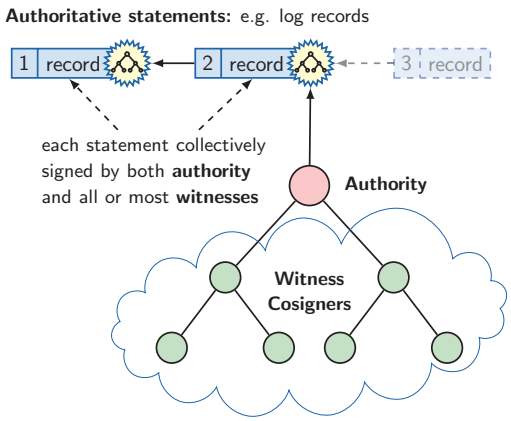

Figure 1: CoSi protocol architecture

mitted and to learn its outcome, making secure transaction *verification* unscalable.

2.3 Scalable Collective Signing

CoSi [54] is a protocol for scalable collective signing, which enables an authority or *leader* to request that statements be publicly validated and *(co-)signed* by a decentralized group of *witnesses*. Each protocol run yields a *collective signature* having size and verification cost comparable to an individual signature, but which compactly attests that both the leader and its (perhaps many) witnesses observed and agreed to sign the statement.

To achieve scalability, CoSi combines Schnorr multi-signatures [51] with communication trees that are long used in multicast protocols [13, 21, 55]. Initially, the protocol assumes that signature verifiers know the public keys of the leader and those of its witnesses, all of which combine to form a well-known aggregate public key. For each message to be collectively signed, the leader then initiates a CoSi four-phase protocol round that require two round-trips over the communication tree between the leader and its witnesses:

1. **Announcement:** The leader broadcasts an announcement of a new round down the communication tree. The announcement can optionally include the message M to be signed, otherwise M is sent in phase three.
2. **Commitment:** Each node picks a random secret and uses it to compute a Schnorr commitment. In a bottom-up process, each node obtains an aggregate Schnorr commitment from its immediate children, combines those with its own commitment, and passes a further-aggregated commitment up the tree.
3. **Challenge:** The leader computes a collective Schnorr challenge using a cryptographic hash function and broadcasts it down the communication tree, along with the message M to sign, if the latter has not already been sent in phase one.
4. **Response:** Using the collective challenge, all nodes compute an aggregate response in a bottom-up fashion that mirrors the commitment phase.

The result of this four-phase protocol is the production of a standard Schnorr signature that requires about 64 bytes, using the Ed25519 elliptic curve [6], and that anyone can verify against the aggregate public key nearly as efficiently as the verification of an individual signature. Practical caveats apply if some witnesses are offline during the collective signing process: in this case the CoSi protocol can proceed, but the resulting signature grows to include metadata verifiably documenting which witnesses did and did not co-sign. We refer to the CoSi paper for details [54].

3 ByzCoin Design

This section presents ByzCoin with a step-by-step approach, starting from a simple "strawman" combination of PBFT and Bitcoin. From this strawman, we progressively address the challenges of determining consensus group membership, adapting Bitcoin incentives and mining rewards, making the PBFT protocol scale to large groups and handling block conflicts and selfish mining.

3.1 System Model

ByzCoin is designed for untrustworthy networks that can arbitrarily delay, drop, re-order or duplicate messages. To avoid the FLP impossibility [27], we assume the network has a weak synchrony property [14]. The ByzCoin system is comprised of a set of N block miners that can generate key-pairs, but there is no trusted public-key infrastructure. Each node i has a limited amount of *hash power* that corresponds to the maximum number of block-header hashes the node can perform per second.

At any time t a subset of miners $M(t)$ is controlled by a malicious attacker and are considered faulty. Byzantine miners can behave arbitrarily, diverting from the protocol and colluding to attack the system. The remaining honest miners follow the prescribed protocol. We assume that the total hash power of all Byzantine nodes is less than $\frac{1}{4}$ of the system's total hash power at any time, since proof-of-work-based cryptocurrencies become vulnerable to selfish mining attacks by stronger adversaries [25].

3.2 Strawman Design: PBFTCoin

For simplicity, we begin with PBFTCoin, an unrealistically simple protocol that naively combines PBFT with Bitcoin, then gradually refine it into ByzCoin.

For now, we simply assume that a group of $n = 3f + 1$ PBFT replicas, which we call *trustees*, has been fixed and

globally agreed upon upfront, and that at most f of these trustees are faulty. As in PBFT, at any given time, one of these trustees is the *leader*, who proposes transactions and drives the consensus process. These trustees collectively maintain a Bitcoin-like blockchain, collecting transactions from clients and appending them via new blocks, while guaranteeing that only one blockchain history ever exists and that it can never be rolled back or rewritten. Prior work has suggested essentially such a design [17, 19], though without addressing the scalability challenges it creates.

Under these simplifying assumptions, PBFTCoin guarantees safety and liveness, as at most f nodes are faulty and thus the usual BFT security bounds apply. However, the assumption of a fixed group of trustees is unrealistic for Bitcoin-like decentralized cryptocurrencies that permit open membership. Moreover, as PBFT trustees authenticate each other via non-transferable symmetric-key MACs, each trustee must communicate directly with most other trustees in every round, thus yielding $O(n^2)$ communication complexity.

Subsequent sections address these restrictions, transforming PBFTCoin into ByzCoin in four main steps:

1. We use Bitcoin's proof-of-work mechanism to determine consensus groups dynamically while preserving Bitcoin's defense against Sybil attacks.
2. We replace MAC-authenticated direct communication with digital signatures to make authentication transferable and thereby enabling sparser communication patterns that can reduce the normal case communication latency from $O(n^2)$ to $O(n)$.
3. We employ scalable collective signing to reduce per-round communication complexity further to $O(\log n)$ and reduce typical signature verification complexity from $O(n)$ to $O(1)$.
4. We decouple transaction verification from leader election to achieve a higher transaction throughput.

3.3 Opening the Consensus Group

Removing PBFTCoin's assumption of a closed consensus group of trustees presents two conflicting challenges. On the one hand, conventional BFT schemes rely on a well-defined consensus group to guarantee safety and liveness. On the other hand, Sybil attacks [23] can trivially break any open-membership protocol involving security thresholds, such as PBFT's assumption that at most f out of $3f+1$ members are honest.

Bitcoin and many of its variations employ a mechanism already suited to this problem: proof-of-work mining. Only miners who have dedicated resources are allowed to become a member of the consensus group. In refining PBFTCoin, we adapt Bitcoin's proof-of-work mining into a *proof-of-membership* mechanism. This

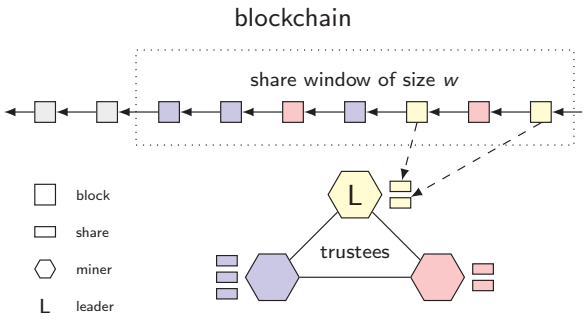

Figure 2: Valid shares for mined blocks in the blockchain are credited to miners

mechanism maintains the "balance of power" within the BFT consensus group over a given fixed-size sliding *share window*. Each time a miner finds a new block, it receives a *consensus group share*, which proves the miner's membership in the group of trustees and moves the share window one step forward. Old shares beyond the current window expire and become useless for purposes of consensus group membership. Miners holding no more valid shares in the current window lose their membership in the consensus group, hence they are no longer allowed to participate in the decision-making.

At a given moment in time, each miner wields "voting power" of a number of shares equal to the number of blocks the miner has successfully mined within the current window. Assuming collective hash power is relatively stable, this implies that within a window, each active miner wields a number of shares statistically proportionate to the amount of hash power that the miner has contributed during this time period.

The size w of the share window is defined by the average block-mining rate over a given time frame and influences certain properties such as the resilience of the protocol to faults. For example, if we assume an average block-mining rate of 10 minutes and set the duration of the time frame to one day (or one week), then $w = 144$ ($w = 1008$). This mechanism limits the membership of miners to recently active ones, which prevents the system from becoming unavailable due to too many trustees becoming inactive over time, or from miners aggregating many shares over an extended period and threatening the balance in the consensus group. The relationship between blocks, miners and shares is illustrated in Fig. 2.

Mining Rewards and Transaction Fees. As we can no longer assume voluntary participation as in PBFTCoin's closed group of trustees, we need an incentive for nodes to obtain shares in the consensus group and to remain active. For this purpose, we adopt Bitcoin's ex-

isting incentives of mining rewards and transaction fees. But instead of these rewards all going to the miner of the most recent block we split a new block's rewards and fees across all members of the current consensus group, in proportion to the number of shares each miner holds. As a consequence, the more hash power a miner has devoted within the current window, hence the more shares the miner holds, the more revenue the miner receives during payouts in the current window. This division of rewards also creates incentives for consensus group members to remain live and participate, because they receive their share of the rewards for new blocks only if they continually participate, in particular contributing to the prepare and commit phases of each BFT consensus round.

3.4 Replacing MACs by Digital Signatures

In our next refinement step towards ByzCoin, we tackle the scalability challenge resulting from PBFT's typical communication complexity of $O(n^2)$, where n is the group size. PBFT's choice of MAC-authenticated all-to-all communication was motivated by the desire to avoid public-key operations on the critical transaction path. However, the cost for public-key operations has decreased due to well-optimized asymmetric cryptosystems [6], making those costs less of an issue.

By adopting digital signatures for authentication, we are able to use sparser and more scalable communication topologies, thus enabling the current leader to collect and distribute third-party verifiable evidence that certain steps in PBFT have succeeded. This removes the necessity for all trustees to communicate directly with each other. With this measure we can either enable the leader to collect and distribute the digital signatures, or let nodes communicate in a chain [32], reducing the normal-case number of messages from $O(n^2)$ to $O(n)$.

3.5 Scalable Collective Signing

Even with signatures providing transferable authentication, the need for the leader to collect and distribute – and for all nodes to verify – many individual signatures per round can still present a scalability bottleneck. Distributing and verifying tens or even a hundred individual signatures per round might be practical. If we want consensus groups with a thousand or more nodes, however (*e.g.*, representing all blocks successfully mined in the past week), it is costly for the leader to distribute 1000 digital signatures and wait for everyone to verify them. To tackle this challenge, we build on the CoSi protocol [54] for collective signing. CoSi does not directly implement consensus or BFT, but it offers a primitive that the leader in a BFT protocol can use to collect and aggregate prepare and commit messages during PBFT rounds.

We implement a single ByzCoin round by using two sequential CoSi rounds initiated by the current leader (*i.e.*, the owner of the current view). The leader's announcement of the first CoSi round (phase 1 in Section 2.3) implements the *pre-prepare* phase in the standard PBFT protocol (Section 2.2). The collective signature resulting from this first CoSi round implements the PBFT protocol's *prepare* phase, in which the leader obtains attestations from a two-thirds super-majority quorum of consensus group members that the leader's proposal is safe and consistent with all previously-committed history.

As in PBFT, this prepare phase ensures that a proposal *can be* committed consistently, but by itself it is insufficient to ensure that the proposal *will be* committed. The leader and/or some number of other members could fail before a super-majority of nodes learn about the successful prepare phase. The ByzCoin leader therefore initiates a second CoSi round to implement the PBFT protocol's *commit* phase, in which the leader obtains attestations from a two-thirds super-majority that all the signing members witnessed the successful result of the prepare phase and made a positive commitment to remember the decision. This collective signature, resulting from this second CoSi round, effectively attests that a two-thirds super-majority of members not only considers the leader's proposal "safe" but promises to remember it, indicating that the leader's proposal is fully committed.

In cryptocurrency terms, the collective signature resulting from the prepare phase provides a proof-of-acceptance of a proposed block of transactions. This block is not yet committed, however, since a Byzantine leader that does not publish the accepted block could double-spend by proposing a conflicting block in the next round. In the second CoSi commit round, the leader announces the proof-of-acceptance to all members, who then validate it and collectively sign the block's hash to produce a collective commit signature on the block. This way a Byzantine leader cannot rewrite history or double-spend, because by counting arguments at least one honest node would have to sign the commit phase of both histories, which an honest node by definition would not do.

The use of CoSi does not affect the fundamental principles or semantics of PBFT but improves its scalability and efficiency in two main ways. First, during the commit round where each consensus group member must verify that a super-majority of members have signed the prior prepare phase, each participant generally needs to receive only an $O(1)$-size rather than $O(n)$-size message, and to expend only $O(1)$ rather than $O(n)$ computation effort by verifying a single collective signature instead of n individual ones. This benefit directly increases the scalability and reduces the bandwidth and computation costs of consensus rounds themselves.

A second benefit is that after the final CoSi commit round has completed, the final resulting collective commit signature serves as a typically $O(1)$-size proof, which anyone can verify in $O(1)$ computation time that a given block – hence any transaction within that block – has been irreversibly committed. This secondary scalability-benefit might in practice be more important than the first, because it enables "light clients" who neither mine blocks nor store the entire blockchain history to verify quickly and efficiently that a transaction has committed, without requiring active communication with or having to trust any particular full node.

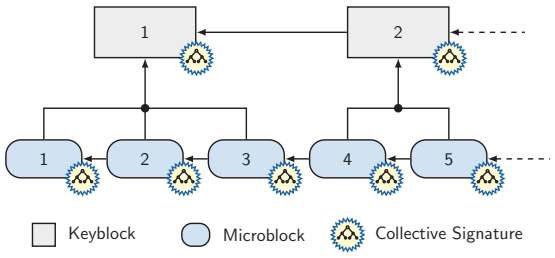

Figure 3: ByzCoin blockchain: Two parallel chains store information about the leaders (keyblocks) and the transactions (microblocks)

3.6 Decoupling Transaction Verification from Leader Election

Although ByzCoin so far provides a scalable guarantee of strong consistency, thus ensuring that clients need to wait only for the next block rather than the next several blocks to verify that a transaction has committed, the time they still have to wait *between* blocks can, nevertheless, be significant: *e.g.*, up to 10 minutes using Bitcoin's difficulty tuning scheme. Whereas ByzCoin's strong consistency might in principle make it "safe" from a consistency perspective to increase block mining rate, doing so could still exacerbate liveness and other performance issues, as in Bitcoin [47]. To enable lower client-perceived transaction latency, therefore, we build on the idea of Bitcoin-NG [24] to decouple the functions of transaction verification from block mining for leader election and consensus group membership.

As in Bitcoin-NG, we use two different kinds of blocks. The first, *microblocks* or transaction blocks, represent transactions to be stored and committed. The current leader creates a new microblock every few seconds, depending on the size of the block, and uses the CoSi-based PBFT protocol above to commit and collectively sign it. The other type of block, *keyblocks*, are mined via proof-of-work as in Bitcoin and serve to elect leaders and create shares in ByzCoin's group membership protocol as discussed earlier in Section 3.3. As in Bitcoin-NG, this decoupling allows the current leader to propose and commit many microblocks that contain many smaller batches of transactions, within one $\approx$ 10-minute keyblock mining period. Unlike Bitcoin-NG, in which a malicious leader could rewrite history or double-spend within this period until the next keyblock, ByzCoin ensures that each microblock is irreversibly committed regardless of the current leader's behavior.

In Bitcoin-NG one blockchain includes both types of blocks, which introduces a race condition for miners. As microblocks are created, the miners have to change the header of their keyblocks to mine on top of the latest microblock. In ByzCoin, in contrast, the blockchain becomes two separate parallel blockchains, as shown in Fig. 3. The main blockchain is the keyblock chain, consisting of all mined blocks. The microblock chain is a secondary blockchain that depends on the primary to identify the era in which every microblock belongs to, *i.e.*, which miners are authoritative to sign it and who is the leader of the era.

Microblocks. A microblock is a simple block that the current consensus group produces every few seconds to represent newly-committed transactions. Each microblock includes a set of transactions and a collective signature. Each microblock also includes hashes referring to the previous microblock and keyblock: the former to ensure total ordering, and the latter indicating which consensus group window and leader created the microblock's signature. The microblock's hash is collectively signed by the corresponding consensus group.

Keyblocks. Each keyblock contains a proof-of-work, which is used to determine consensus group membership via the sliding-window mechanism discussed earlier, and to pay signers their rewards. Each newly-mined keyblock defines a new consensus group, and hence a new set of public keys with which the next era's microblocks will be collectively signed. Since each successive consensus group differs from the last in at most one member, PBFT ensures the microblock chain's consistency and continuity across this group membership change provided at most f out of $3f+2$ members are faulty.

Bitcoin-NG relies on incentives to discourage the next leader from accidentally or maliciously "forgetting" a prior leader's microblocks. In contrast, the honest supermajority in a ByzCoin consensus group will refuse to allow a malicious or amnesiac leader to extend any but the most recently-committed microblock, regardless of which (current or previous) consensus group committed it. Thus, although competing keyblock conflicts may still appear, these "forks" cannot yield an inconsistent microblock chain. Instead, a keyblock conflict can at

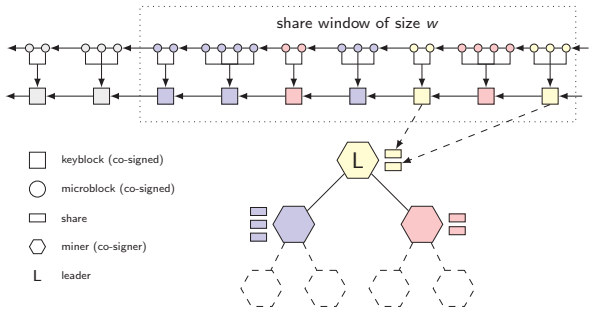

Figure 4: Overview of the full ByzCoin design

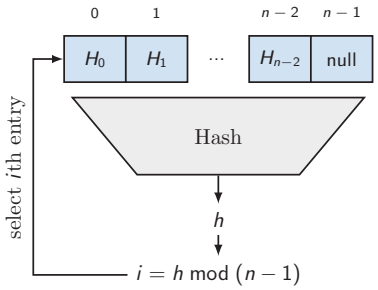

Figure 5: Deterministic fork resolution in ByzCoin

worst temporarily interrupt the PBFT protocol's liveness, until it is resolved as mentioned in Section 3.6.1.

Decoupling transacton verification from leader election and consensus group evolution in this way brings the overall ByzCoin architecture to completion, as illustrated in Fig. 4. Subsequent sections discuss further implications and challenges this architecture presents.

3.6.1 Keyblock Conflicts and Selfish Mining

PBFT's strong consistency by definition does not permit inconsistencies such as forks in the microblock chain. The way the miners collectively decide how to resolve keyblock conflicts, however, can still allow selfish mining [25] to occur as in Bitcoin. Worse, if the miners decide randomly to follow one of the two blocks, then keyblock forks might frequently lead to PBFT liveness interruptions as discussed above, by splitting miners "too evenly" between competing keyblocks. Another approach to deciding between competing keyblocks is to impose a deterministic priority function on their hash values, such as "smallest hash wins." Unfortunately, this practice can encourage selfish mining.

One way to break a tie without helping selfish miners, is to increase the entropy of the output of the deterministic prioritization function. We implement this idea using the following algorithm. When a miner detects a keyblock fork, it places all competing blocks' header hashes into a sorted array, from low to high hash values. The array itself is then hashed, and the final bit(s) of this hash determine which keyblock wins the fork.

This solution, shown in Fig. 5, also uses the idea of a deterministic function applied to the blocks, thus requiring no voting. Its advantage is that the input of the hash function is partially unknown before the fork occurs, thus the entropy of the output is high and difficult for an attacker to be able to optimize. Given that the search space for a possible conflict is as big as the search space for a new block, trying to decide if a block has better than 50% probability of winning the fork is as hard as finding a new block.

3.6.2 Leader Election and PBFT View Changes

Decoupling transaction verification from the block-mining process comes at a cost. So far we have assumed, as in PBFT, that the leader remains fixed unless he fails. If we keep this assumption, then this leader gains the power of deciding which transactions are verified, hence we forfeit the fairness-enforcing benefit of Bitcoin's leader election. To resolve this issue, every time a keyblock is signed, ByzCoin forces a mandatory PBFT view-change to the keyblock's miner. This way the power of verifying transactions in blocks is assigned to the rightful miner, who has an era of microblock creation from the moment his keyblock is signed until the next keyblock is signed.

When a keyblock conflict occurs, more than one such "mandatory" view-change occurs, with the successful miners trying to convince other participants to adopt their keyblock and its associated consensus group. For example, in a keyblock fork, one of the two competing keyblocks wins the resolution algorithm described above. However, if the miner of the "losing" block races to broadcast its keyblock and more than 33% honest miners have already committed to it before learning about the competing keyblock, then the "winning" miner is too late and the system either commits to the first block or (in the worst case) loses liveness temporarily as discussed above. This occurs because already-committed miners will not accept a mandatory view-change except to a keyblock that represents their committed state and whose microblock chain extends all previously-committed microblocks. Further analysis of how linearizability is preserved across view-changes may be found in the original PBFT paper [14].

3.6.3 Tree Creation in ByzCoin

Once a miner successfully creates a new keyblock, he needs to form a CoSi communication tree for collective signing, with himself as the leader. If all miners individually acknowledge this keyblock to transition to

the next view, this coordination normally requires $O(N)$ messages. To avoid this overhead at the beginning of each keyblock round, the miners autonomously create the next round's tree bottom-up during the previous keyblock round. This can be done in $O(1)$ by using the blockchain as an array that represents a full tree.

This tree-building process has three useful side-effects. First, the previous leader is the first to get the new block, hence he stops creating microblocks and wasting resources by trying to sign them. Second, in the case of a keyblock conflict, potential leaders use the same tree, and the propagation pattern is the same; this means that all nodes will learn and decide on the conflict quickly. Finally, in the case of a view change, the new view will be the last view that worked correctly. So if the leader of the keyblock i fails, the next leader will again be the miner of keyblock $i-1$.

3.7 Tolerating Churn and Byzantine Faults

In this section we discuss the challenges of fault tolerance in ByzCoin, particularly tree failures and maximum tolerance for Byzantine faults.

3.7.1 Tree Fault Tolerance

In CoSi, there are multiple different mechanisms that allow substantial fault-tolerance. Furthermore the strict membership requirements and the frequent tree changes of ByzCoin increase the difficulty for a malicious attacker with less than around 25% of the total hash power to compromise the system. A security analysis, however, must assume that a Byzantine adversary is able to get the correct nodes of the ByzCoin signing tree so that it can compromise the liveness of the system by a simple DoS.

To mitigate this risk, we focus on recent Byzantine fault tolerance results [32], modifying ByzCoin so that the tree communication pattern is a normal-case performance optimization that can withstand most malicious attacks. But when the liveness of the tree-based ByzCoin is compromised, the leader can return to non-tree-based communication until the end of his era.

The leader detects that the tree has failed with the following mechanism: After sending the block to his children, the leader starts a timer that expires before the view-change timer. Then he broadcasts the hash of the block he proposed and waits. When the nodes receive this message they check if they have seen the block and either send an ACK or wait until they see the block and then send the ACK. The leader collects and counts the ACKs, to detect if his block is rejected simply because it never reaches the witnesses. If the timer expires or a block rejection arrives before he receives two-thirds of the ACKs, the leader knows that the tree has failed and reverts to a flat ByzCoin structure before the witnesses assume that he is faulty.

As we show in Section 4, the flat ByzCoin structure can still quickly sign keyblocks for the day-long window (144 witnesses) while maintaining a throughput higher than Bitcoin currently supports. Flat ByzCoin is more robust to faults, but increases the communication latency back to $O(n)$. Furthermore, if all faults ($\lfloor \frac{N}{3} \rfloor$) are consecutive leaders, this can lead back to a worst case $O(n^2)$ communication latency.

3.7.2 Membership Churn and BFT

After a new leader is elected, the system needs to ensure that the first microblock of the new leader points to the last microblock of the previous leader. Having $2f+1$ supporting votes is not enough. This occurs because there is the possibility than an honest node lost its membership when the new era started. Now in the worst case, the system has f Byzantine nodes, f honest nodes that are up to date, f slow nodes that have a stale view of the blockchain, and the new leader that might also have a stale view. This can lead to the leader proposing a new microblock, ignoring some signed microblocks and getting $2f+1$ support (stale+Byzantine+his own). For this reason, the first microblock of an era needs $2f+2$ supporting signatures. If the leader is unable to obtain them, this means that he needs to synchronize with the system, *i.e.*, he needs to find the latest signed microblock from the previous roster. He asks all the nodes in his roster, plus the node that lost its membership, to sign a latest-checkpoint message containing the hash of the last microblock. At this point in time, the system has $3f+2$ ($3f+1$ of the previous roster plus the leader) members and needs $2f+1$ honest nodes to verify the checkpoint, plus an honest leader to accept it (a Byzantine leader will be the $f+1$ fault and compromise liveness). Thus, ByzCoin can tolerate f fails in a total of $3f+2$ nodes.

4 Performance Evaluation

In this section we discuss the evaluation of the ByzCoin prototype and our experimental setup. The main question we want to evaluate is whether ByzCoin is usable in practice without incurring large overheads. In particular we focus on consensus latency and transaction throughput for different parameter combinations.

4.1 Prototype Implementation

We implemented ByzCoin in Go[1] and made it publicly available on GitHub.[2] ByzCoin's consensus mecha-

[1] `https://golang.org`
[2] `https://github.com/DeDiS/Cothority`

nism is based on the CoSi protocol with Ed25519 signatures [6] and implements both flat- and tree-based collective signing layouts as described in Section 3. For comparison, we also implemented a conventional PBFT consensus algorithm with the same communication patterns as above and a consensus algorithm that uses individual signatures and tree-based communication.

To simulate consensus groups of up to 1008 nodes, we oversubscribed the available 36 physical machines (see below) and ran up to 28 separate ByzCoin processes on each server. Realistic wide-area network conditions are mimicked by imposing a round-trip latency of 200 ms between any two machines and a link bandwidth of 35 Mbps per simulated host. Note that this simulates only the connections between miners of the consensus group and not the full Bitcoin network. Full nodes and clients are not part of the consensus process and can retrieve signed blocks only after consensus is reached. Since Bitcoin currently is rather centralized and has only a few dozen mining pools [3], we assume that if/when decentralization happens, all miners will be able to support these rather constrained network requirements.

The experimental data to form microblocks was taken by ByzCoin clients from the actual Bitcoin blockchain. Both micro- and keyblocks are fully transmitted and collectively signed through the tree and are returned to the clients upon request together with the proof. Verification of block headers is implemented but transaction verification is only emulated to avoid further measurement distortion through oversubscribed servers. A similar practice is used in Shadow Bitcoin [45]. We base our emulation both on measurements [31] of the average block-verification delays (around 200 ms for 500 MB blocks) and on the claims of Bitcoin developers [8] that as far as hardware is concerned Bitcoin can easily verify 4000 TPS. We simulate a linear increase of this delay proportional to the number of transactions included in the block. Because of the communication pattern of ByzCoin, the transaction-verification cost delays only the leaf nodes. By the time the leaf nodes finish the block verification and send their vote back to their parents, all other tree nodes should have already finished the verification and can immediately proceed. For this reason the primary delay factor is the transmission of the blocks that needs to be done $\log N$ sequential times.

We ran all our experiments on DeterLab [22] using 36 physical machines, each having four Intel E5-2420 v2 CPUs and 24 GB RAM and being arranged in a star-shaped virtual topology.

4.2 Consensus Latency

The first two experiments focus on how microblock commitment latency scales with consensus group size and with number of transactions per block.

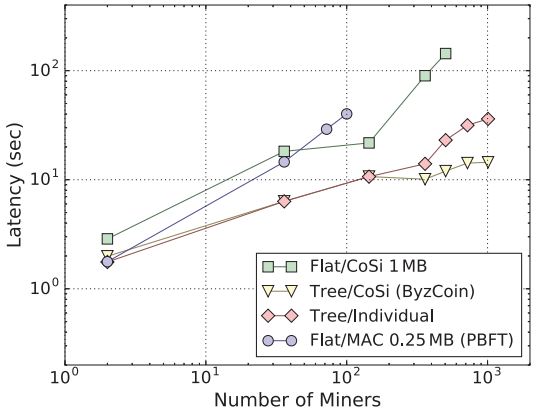

Figure 6: Influence of the consensus group size on the consensus latency

4.2.1 Consensus Group Size Comparison

This experiment focuses on the scalability of ByzCoin's BFT protocol in terms of the consensus group size. The number of unique miners participating in a consensus group is limited by the number of membership shares in the window (Section 3.3), but can be smaller if some miners hold multiple shares (*i.e.*, successfully mined several blocks) within the same window.

We ran experiments for Bitcoin's maximum block size (1 MB) with a variable number of participating hosts. Every time we increased the number of hosts, we also increased the servers' bandwidth so that the available bandwidth per simulated host remained constant (35 Mbps). For the PBFT simulation, the 1 MB block was too big to handle, thus the PBFT line corresponds to a 250 KB block size.

As Fig. 6 shows, the simple version of ByzCoin achieves acceptable latency, as long as the consensus group size is less than 200. After this point the cost for the leader to broadcast the block to everyone incurs large overheads. On the contrary, the tree-based ByzCoin scales well, since the same 1 MB block for 1008 nodes suffers signing latency less than the flat approach for 36 nodes. Adding 28 times more nodes (from 36 to 1008) causes a latency increase close to a factor 2 (from 6.5 to 14 seconds). The basic PBFT implementation is quite fast for 2 nodes but scales poorly (40 seconds for 100 nodes), whereas the tree-based implementation with individual signatures performs the same as ByzCoin for up to 200 hosts. If we aim for the higher security level of 1008 nodes, however, then ByzCoin is 3 times faster.

Fig. 7 shows the performance cost of keyblock sign-

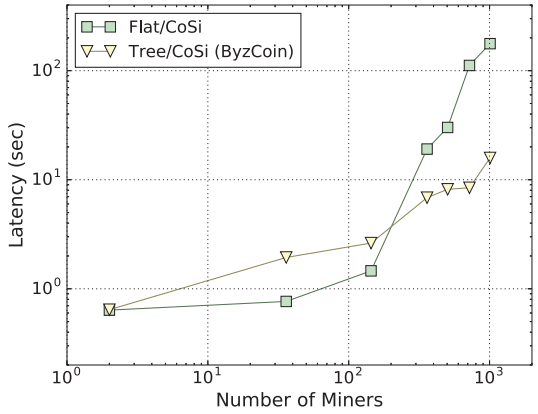

Figure 7: Keyblock signing latency

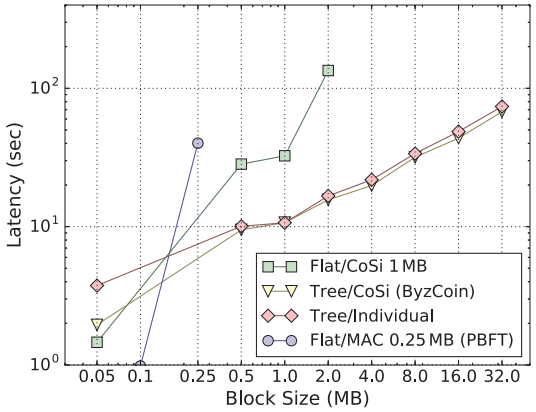

Figure 8: Influence of the block size on the consensus latency

ing. The flat variant outperforms the tree-based version when the number of hosts is small since the blocks have as many transactions as there are hosts and thus are small themselves. This leads to a fast transmission even in the flat case and the main overhead comes from the block propagation latency, which scales with $O(\log N)$ in the tree-based ByzCoin variant.

4.2.2 Block Size Comparison

The next experiment analyzes how different block sizes affect the scalability of ByzCoin. We used a constant number of 144 hosts for all implementations. Once again, PBFT was unable to achieve acceptable latency with 144 nodes, thus we ran it with 100 nodes only.

Fig. 8 shows the average latency of the consensus mechanism, determined over 10 blocks when their respective sizes increase. As in the previous section we see that the flat implementation is acceptable for a 1 MB block, but when the block increases to 2 MB the latency quadruples. This outcome is expected as the leader's link saturates when he tries to send 2 MB messages to every participating node. In contrast ByzCoin scales well because the leader outsources the transmission of the blocks to other nodes and contacts only his children. The same behavior is observed for the algorithm that uses individual signatures and tree-based communication, which shows that the block size has no negative effect on scalability when a tree is used. Finally, we find that PBFT is fast for small blocks, but the latency rapidly increases to 40 seconds for 250 KB blocks.

ByzCoin's signing latency for a 1 MB block is close to 10 seconds, which should be small enough to make the need for 0-confirmation transactions almost disappear. Even for a 32 MB block ($\approx$ 66000 transactions) the delay is much lower (around 90 seconds) than the $\approx$ 10 minutes required by Bitcoin.

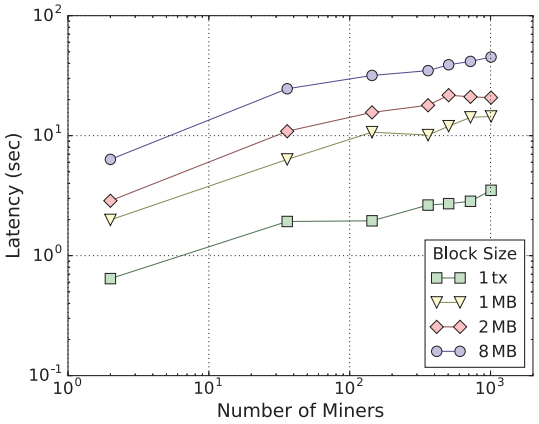

Figure 9: Influence of the consensus group size on the block signing latency

Fig. 9 demonstrates the signing latency of various blocks sizes on tree-based ByzCoin. Signing one-transaction blocks takes only 3 seconds even when 1008 miners co-sign it. For bigger blocks, we have included Bitcoin's current maximum block size of 1 MB along with the proposed limits of 2 MB in Bitcoin Classic and 8 MB in Bitcoin Unlimited [2]. As the graph shows, 1 MB and 2 MB blocks scale linearly in number of nodes at first but after 200 nodes, the propagation latency is higher than the transmission of the block, hence the latency is close to constant. For 8 MB blocks, even with 1008 the signing latency increases only linearly.

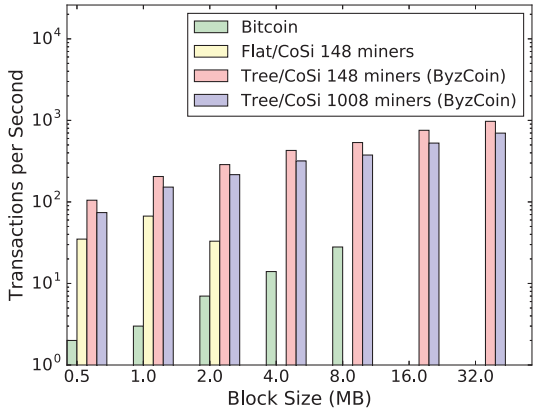

Figure 10: Throughput of ByzCoin

4.3 Transaction Throughput

In this experiment, we investigate the maximum throughput in terms of transactions per second (TPS) that ByzCoin can achieve, and show how Bitcoin could improve its throughput by adopting a ByzCoin-like deployment model. We tested ByzCoin versions with consensus group sizes of 144 and 1008 nodes, respectively. Note that performance-wise this resembles the worst case scenario since the miner-share ratio is usually not 1:1 as miners in the consensus group are allowed to hold multiple shares, as described in Section 3.3.

Analyzing Fig. 10 shows that Bitcoin can increase its overall throughput by more than one order of magnitude through adoption of a flat ByzCoin-like model, which separates transaction verification and block mining and deals with forks via strong consistency. Furthermore, the 144 node configuration can achieve close to 1000 TPS, which is double the throughput of Paypal, and even the 1008-node roster achieves close to 700 TPS. Even when the tree fails, the system can revert back to 1 MB microblocks on the flat and more robust variant of ByzCoin and still have a throughput ten times higher than the current maximum throughput of Bitcoin.

In both Figs. 8 and 10, the usual trade-off between throughput and latency appears. The system can work with 1–2 MB microblocks when the load is normal and then has a latency of 10–20 seconds. If an overload occurs, the system adaptively changes the block size to enable higher throughput. We note that this is not the case in the simple ByzCoin where 1 MB microblocks have optimal throughput and acceptable latency.

5 Security Analysis

In this section, we conduct a preliminary, informal security analysis of ByzCoin, and discuss how its consensus mechanism can mitigate or eliminate some known attacks against Bitcoin.

5.1 Transaction Safety

In the original Bitcoin paper [47], Nakamoto models Bitcoin's security against transaction double spending attacks as in a Gambler's Ruin Problem. Furthermore, he models the progress an attacker can make as a Poisson distribution and combines these two models to reach Eq. (1). This equation calculates the probability of a successful double spend after z blocks when the adversary controls q computing power.

$$P = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - \left(\frac{q}{p}\right)^{(z-k)}\right) \qquad (1)$$

In Figs. 11 and 12 we compare the relative safety of a transaction over time in Bitcoin[3] versus ByzCoin. Fig. 11 shows that ByzCoin can secure a transaction in less than a minute, because the collective signature guarantees forward security. On the contrary, Bitcoin's transactions need hours to be considered fully secured from a double-spending attempt. Fig. 12 illustrates the required time from transaction creation to the point where a double spending attack has less than 0.1% chance of success. ByzCoin incurs a latency of below one minute to achieve the above security, which boils down to the time the systems needs to produce a collectively signed microblock. Bitcoin on the other hand needs several hours to reach the same guarantees. Note that this graph does not consider other advanced attacks, such as eclipse attacks [34], where Bitcoin offers no security for the victim's transactions.

5.2 Proof-of-Membership Security

The security of ByzCoin's proof-of-membership mechanism can be modeled as a random sampling problem with two possible independent outcomes (honest, Byzantine). The probability of picking a Byzantine node (in the worst case) is $p = 0.25$ and the number of tries corresponds to the share window size w. In this setting, we are interested in the probability that the system picks less than $c = \lfloor \frac{w}{3} \rfloor$ Byzantine nodes as consensus group members and hence guarantees safety. To calculate this probability, we use the cumulative binomial distribution where X is the random variable that represents the number of times we pick a Byzantine node:

[3] Based on data from `https://blockchain.info`.

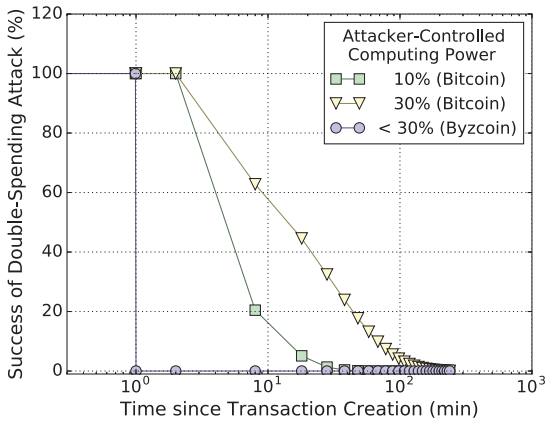

Figure 11: Successful double-spending attack probability

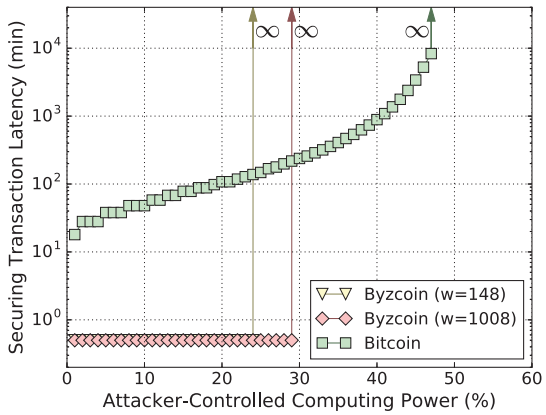

Figure 12: Client-perceived secure transaction latency

$$P[X \leq c] = \sum_{k=0}^{c} \binom{w}{k} p^k (1-p)^{w-k} \qquad (2)$$

Table 1 displays the results for the evaluation of Eq. (2) for various window sizes w both in the common threat model where an adversary controls up to 25% hash power and in the situation where the system faces a stronger adversary with up to 30% computing power. The latter might temporarily occur due to hash power variations and resource churn.

Table 1: Expected proof-of-membership security levels

$p \mid w$	12	100	144	288	1008	2016
0.25	0.842	0.972	0.990	0.999	0.999	1.000
0.30	0.723	0.779	0.832	0.902	0.989	0.999

At this point, recall that w specifies the number of available shares and not necessarily the number of actual miners as each member of the consensus group is allowed to hold multiple shares. This means that the number of available shares gives an upper bound on the latency of the consensus mechanism with the worst case being that each member holds exactly one share.

In order to choose a value for w appropriately one must take into account not only consensus latency and the desired security level (ideally $\geq 99\%$) but also the increased chance for resource churn when values of w become large. From a security perspective the results of Table 1 suggest that the share window size should not be set to values lower than $w = 144$. Ideally, values of $w = 288$ and above should be chosen to obtain a reasonable security margin and, as demonstrated in Section 4, values up to $w = 1008$ provide excellent performance numbers.

Finally, care should be taken when bootstrapping the protocol, as for small values of w there is a high probability that a malicious adversary is able to take over control. For this reason we suggest that ByzCoin starts with vanilla Nakamoto consensus and only after w keyblocks are mined the ByzCoin consensus is activated.

5.3 Defense Against Bitcoin Attacks

0-confirmation Double-Spend Attacks. Race [35] and Finney [26] attacks belong to the family of 0-confirmation double-spend attacks which might affect traders that provide real-time services to their clients. In such scenarios the time between exchange of currency and goods is usually short because traders often cannot afford to wait an extended period of time (10 or more minutes) until a transaction they received can be considered indeed confirmed.

ByzCoin can mitigate both attacks by putting the merchant's transaction in a collectively signed microblock whose verification latency is in the order of a few seconds up to a minute. If this latency is also unacceptable, then he can send a single transaction for signing, which will cost more, but is secured in less than 4 seconds.

N-confirmation Double-Spend Attacks. The assumption underlying this family of attacks [7] is that, after receiving a transaction for a trade, a merchant waits $N-1$ additional blocks until he concludes the interaction with his client. At this point, a malicious client creates a new double-spending transaction and tries to fork the blockchain, which has a non-negligible success-probability if the adversary has enough hash power. For example, if $N = 3$ then an adversary holding 10% of the network's hash power has a 5% success-chance to mount the above attack [47].

In ByzCoin the merchant would simply check the collective signature of the microblock that includes the

transaction, which allows him to verify that it was accepted by a super-majority of the network. Afterwards the attacker cannot succeed in forking the blockchain as the rest of the signers will not accept his new block. Even if the attacker is the leader, the proposed microblock will be rejected, and a view change will occur.

Eclipse and Delivery-Tampering Attacks. In an eclipse attack [34] it is assumed that an adversary controls a sufficiently large number of connections between the victim and the Bitcoin network. This enables the attacker to mount attacks such as 0- and N-confirmation double-spends with an ever increasing chance of success the longer the adversary manages to keep his control over the network. Delivery-tampering attacks [31] exploit Bitcoin's scalability measures to delay propagation of blocks without causing a network partition. This allows an adversary to control information that the victim receives and simplifies to mount 0- and 1-confirmation double-spend attacks as well as selfish-mining.

While ByzCoin does not prevent an attacker from eclipsing a victim or delaying messages in the peer-to-peer network, its use of collective signatures in transaction commitment ensure that a victim cannot be tricked into accepting an alternate attacker-controlled transaction history produced in a partitioned network fragment.

Selfish and Stubborn Mining Attacks. Selfish mining [25] allows a miner to increase his profit by adding newly mined blocks to a hidden blockchain instead of instantly broadcasting them. This effect can be further amplified if the malicious miner has good connectivity to the Bitcoin network. The authors of selfish mining propose a countermeasure that thwarts the attack if a miner has less than 25% hash power under normal circumstances or less than 33% in case of an optimal network. Stubborn mining [48] further generalizes the ideas behind selfish mining and combines it with eclipse attacks in order to increase the adversary's revenue.

In ByzCoin, these strategies are ineffective as forks are instantly resolved in a deterministic manner, hence building a hidden blockchain only wastes resources and minimizes revenue. Another approach to prevent the above attacks would be to include bias-resistant public randomness [40] in every keyblock. This way even if an attacker gains control over the consensus mechanism (*e.g.*, by having $> 33\%$ hash power) he would still be unable to mine hidden blocks. We leave exploring this approach for future research.

Transaction Censorship. In Bitcoin-NG, a malicious leader can censor transactions for the duration of his epoch(s). The same applies for ByzCoin. However, as not every leader is malicious, the censored transactions are only delayed and will be processed eventually by the next honest leader. ByzCoin can improve on this, as the leader's actions are double-checked by all the other miners in the consensus group. A client can announce his censored transaction just like in classic PBFT; this will indicate a potential leader fault and will either stop censorship efforts or lead to a view-change to remove the malicious leader. Finally, in Bitcoin(-NG) a miner can announce his intention to fork over a block that includes a transaction, giving an incentive to other miners to exclude this transaction. In ByzCoin using fork-based attacks to censor transactions is no longer possible due to ByzCoin's deterministic fork resolution mechanism. An attacker can therefore only vote down a leader's proposals by refusing to co-sign. This is also a limitation, however, as an adversary who controls more than 33% of the shares (Section 7) deny service and can censor transactions for as long as he wants.

6 Related Work

ByzCoin and Bitcoin [47] share the same primary objective: implement a state machine replication (SMR) system with open membership [9, 29]. Both therefore differ from more classic approaches to Byzantine fault-tolerant SMRs with static or slowly changing consensus groups such as PBFT [14], Tendermint [10], or Hyperledger [42].

Bitcoin has well-known performance shortcomings; there are several proposals [41, 57] on how to address these. The GHOST protocol [53] changes the chain selection rule when a fork occurs. While Bitcoin declares the fork with the most proof-of-work as the new valid chain, GHOST instead chooses the entire subtree that received the most computational effort. Put differently, the subtree that was considered correct for the longest time will have a high possibility of being selected, making an intentional fork much harder. One limitation of GHOST is that no node will know the full tree, as invalid blocks are not propagated. While all blocks could be propagated, this makes the system vulnerable to DoS attacks since an adversary can simply flood the network with low-difficulty blocks.

Off-chain transactions, an idea that originated from the two-point channel protocol [33], are another alternative to improve latency and throughput of the Bitcoin network. Other similar proposals include the Bitcoin Lightning Network [50] and micro-payment channels [20], which allow transactions without a trusted middleman. They use contracts so that any party can generate proof-of-fraud on the main blockchain and thereby deny revenue to an attacker. Although these systems enable faster cryptocurrencies, they do not address the core problem

of scaling SMR systems, thus sacrificing the open and distributed nature of Bitcoin. Finally, the idea behind sidechains [5] is to connect multiple chains with each other and enable the transfer of Bitcoins from one chain to another. This enables the workload distribution to multiple subsets of nodes that run the same protocol.

There are several proposals that, like ByzCoin, target the consensus mechanism and try to improve different aspects. Ripple [52] implements and runs a variant of PBFT that is low-latency and based on collectively-trusted subnetworks with slow membership changes. The degree of decentralization depends on the concrete configuration of an instance. Tendermint [10] also implements a PBFT-like algorithm, but evaluates it with at most 64 "validators". Furthermore, Tendermint does not address important challenges such as the link-bandwidth between validators, which we found to be a primary bottleneck. PeerCensus [19] is an interesting alternative that shares similarities with ByzCoin, but is only a preliminary theoretical analysis.

Finally, Stellar [43] proposes a novel consensus protocol named Federated Byzantine Agreement, which introduces Quorum slices that enable a BFT protocol "open for anyone to participate". Its security, however, depends on a nontrivial and unfamiliar trust model requiring correct configuration of trustees by each client.

7 Limitations and Future Work

This section briefly outlines several of ByzCoin's important remaining limitations, and areas for future work.

Consensus-Group Exclusion. A malicious ByzCoin leader can potentially exclude nodes from the consensus process. This is easier in the flat variant, where the leader is responsible for contacting every participating miner, but it is also possible in the tree-based version, if the leader "reorganizes" the tree and puts nodes targeted for exclusion in subtrees where the roots are colluding nodes. A malicious leader faces a dilemma, though: excluded nodes lose their share of newly minted coins which increases the overall value per coin and thus the leader's reward. The victims, however, will quickly broadcast view-change messages in an attempt to remove the Byzantine leader.

As an additional countermeasure to mitigate such an attack, miners could run a peer-to-peer network on top of the tree to communicate protocol details. Thus each node potentially receives information from multiple sources. If the parent of a node fails to deliver the announcement message of a new round, this node could then choose to attach itself (together with its entire subtree) to another participating (honest) miner. This self-adapting tree could mitigate the leader's effort to exclude miners. As a last resort, the malicious leader could exclude the commitments of the victims from the aggregate commitment, but as parts of the tree have witnessed these commitments, the risk of triggering a view-change is high.

In summary, the above attack seems irrational as the drawbacks of trying to exclude miners seem to outweigh the benefits. We leave a more thorough analysis of this situation for future work.

Defenses Against 33%+ Attacks. An attacker powerful enough to control more than $\frac{1}{3}$ of the consensus shares can, in the Byzantine threat model, trivially censor transactions by withholding votes, and double-spend by splitting honest nodes in two disjoint groups and collecting enough signatures for two conflicting microblocks. Fig. 12 shows how the safety of ByzCoin fails at 30%, whereas Bitcoin remains safe even for 48%, if a client can wait long enough.

However, the assumption that an attacker completely controls the network is rather unrealistic, especially if messages are authenticated and spoofing is impossible [3]. The existence of the peer-to-peer network on top of the tree, mentioned in the previous paragraph, enables the detection of equivocation attacks such as microblock forks and mitigates the double-spending efforts, as honest nodes will stop following the leader. Thus, double-spending and history rewriting attacks in ByzCoin become trivial only after the attacker has 66% of the shares, effectively increasing the threshold from 51% to 66%. This assumption is realistic, as an attacker controlling the complete network can actually split Bitcoin's network in two halves and trivially double-spend on the weaker side. This is possible because the weak side creates blocks that will be orphaned once the partition heals. We again leave a more thorough analysis of this situation for future work.

Proof-of-Work Alternatives. Bitcoin's hash-based proof-of-work has many drawbacks, such as energy waste and the efficiency advantages of custom ASICs that have made mining by "normal users" impractical. Many promising alternatives are available, such as memory-intensive puzzles [4], or proof-of-stake designs [37]. Consensus group membership might in principle also be based on other Sybil attack-resistant methods, such as those based on social trust networks [58]. A more democratic alternative might be to apportion mining power on a "1 person, 1 vote" principle, based on anonymous *proof-of-personhood* tokens distributed at pseudonym parties [28]. Regardless, we treat the ideal choice of Sybil attack-resistance mechanism as an issue for future work, orthogonal to the focus of this paper.

Other Directions. Besides the issues outlined above, there are many more interesting open questions worth considering: Sharding [17] presents a promising approach to scale distributed protocols and was already studied for private blockchains [18]. A sharded variant of ByzCoin might thus achieve even better scalability and performance numbers. A key obstacle that needs to be analyzed in that context before though is the generation of bias-resistant public randomness [40] which would enable to pick members of a shard in a distributed and secure manner. Yet another challenge is to find ways to increase incentives of rational miners to remain honest, like binding coins and destroying them when misbehavior is detected [10]. Finally, asynchronous BFT [12, 11] is another interesting class of protocols, which only recently started to be analyzed in the context of blockchains [46].

8 Conclusion

ByzCoin is a scalable Byzantine fault tolerant consensus algorithm for open decentralized blockchain systems such as Bitcoin. ByzCoin's strong consistency increases Bitcoin's core security guarantees—shielding against attacks on the consensus and mining system such as N-confirmation double-spending, intentional blockchain forks, and selfish mining—and also enables high scalability and low transaction latency. ByzCoin's application to Bitcoin is just one example, though: theoretically, it can be deployed to any blockchain-based system, and the proof-of-work-based leader election mechanism might be changed to another approach such as proof-of-stake. If open membership is not an objective, the consensus group could be static, though still potentially large. We developed a wide-scale prototype implementation of ByzCoin, validated its efficiency with measurements and experiments, and have shown that Bitcoin can increase the capacity of transactions it handles by more than two orders of magnitude.

Acknowledgments

We would like to thank the DeterLab project team for providing the infrastructure for our experimental evaluation, Joseph Bonneau for his input on our preliminary design, and the anonymous reviewers for their helpful feedback.

References

[1] Abd-El-Malek, M., Ganger, G. R., Goodson, G. R., Reiter, M. K., and Wylie, J. J. Fault-scalable Byzantine Fault-tolerant Services. *SIGOPS Operating Systems Review 39*, 5 (Oct. 2005), 59–74.

[2] Andresen, G. Classic? Unlimited? XT? Core?, Jan. 2016.

[3] Apostolaki, M., Zohar, A., and Vanbever, L. Hijacking Bitcoin: Large-scale Network Attacks on Cryptocurrencies. *arXiv preprint arXiv:1605.07524* (2016).

[4] Ateniese, G., Bonacina, I., Faonio, A., and Galesi, N. Proofs of Space: When Space is of the Essence. In *Security and Cryptography for Networks*. Springer, 2014, pp. 538–557.

[5] Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., and Wuille, P. Enabling Blockchain Innovations with Pegged Sidechains.

[6] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. High-speed high-security signatures. *Journal of Cryptographic Engineering 2*, 2 (2012), 77–89.

[7] Bitcoin Wiki. Confirmation, 2016.

[8] Bitcoin Wiki. Scalability, 2016.

[9] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J., and Felten, E. W. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy. IEEE* (2015).

[10] Buchman, E. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains, 2016.

[11] Cachin, C., Kursawe, K., Petzold, F., and Shoup, V. Secure and Efficient Asynchronous Broadcast Protocols. In *Advances in Cryptology (CRYPTO)* (Aug. 2001).

[12] Cachin, C., Kursawe, K., and Shoup, V. Random Oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. In *19th ACM Symposium on Principles of Distributed Computing (PODC)* (July 2000).

[13] Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., and Singh, A. SplitStream: high-bandwidth multicast in cooperative environments. In *ACM Symposium on Operating Systems Principles (SOSP)* (2003).

[14] CASTRO, M., AND LISKOV, B. Practical Byzantine Fault Tolerance. In *3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (Feb. 1999).

[15] CLEMENT, A., WONG, E. L., ALVISI, L., DAHLIN, M., AND MARCHETTI, M. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In *6th USENIX Symposium on Networked Systems Design and Implementation* (Apr. 2009).

[16] COWLING, J., MYERS, D., LISKOV, B., RODRIGUES, R., AND SHRIRA, L. HQ Replication: A Hybrid Quorum Protocol for Byzantine Fault Tolerance. In *7th Symposium on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2006), OSDI '06, USENIX Association, pp. 177–190.

[17] CROMAN, K., DECKE, C., EYAL, I., GENCER, A. E., JUELS, A., KOSBA, A., MILLER, A., SAXENA, P., SHI, E., SIRER, E. G., AN, D. S., AND WATTENHOFER, R. On Scaling Decentralized Blockchains (A Position Paper). In *3rd Workshop on Bitcoin and Blockchain Research* (2016).

[18] DANEZIS, G., AND MEIKLEJOHN, S. Centrally Banked Cryptocurrencies.

[19] DECKER, C., SEIDEL, J., AND WATTENHOFER, R. Bitcoin Meets Strong Consistency. In *17th International Conference on Distributed Computing and Networking (ICDCN), Singapore* (January 2016).

[20] DECKER, C., AND WATTENHOFER, R. A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. In *Stabilization, Safety, and Security of Distributed Systems*. Springer, Aug. 2015, pp. 3–18.

[21] DEERING, S. E., AND CHERITON, D. R. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems 8*, 2 (May 1990).

[22] DeterLab Network Security Testbed, September 2012.

[23] DOUCEUR, J. R. The Sybil Attack. In *1st International Workshop on Peer-to-Peer Systems (IPTPS)* (Mar. 2002).

[24] EYAL, I., GENCER, A. E., SIRER, E. G., AND VAN RENESSE, R. Bitcoin-NG: A Scalable Blockchain Protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)* (Santa Clara, CA, Mar. 2016), USENIX Association.

[25] EYAL, I., AND SIRER, E. G. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454.

[26] FINNEY, H. Best practice for fast transaction acceptance – how high is the risk?, Feb. 2011. Bitcoin Forum comment.

[27] FISCHER, M. J., LYNCH, N. A., AND PATERSON, M. S. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM) 32*, 2 (1985), 374–382.

[28] FORD, B., AND STRAUSS, J. An offline foundation for online accountable pseudonyms. In *1st International Workshop on Social Network Systems (SocialNets)* (2008).

[29] GARAY, J., KIAYIAS, A., AND LEONARDOS, N. The Bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT 2015*. Springer, 2015, pp. 281–310.

[30] GERVAIS, A., KARAME, G. O., WUST, K., GLYKANTZIS, V., RITZDORF, H., AND CAPKUN, S. On the Security and Performance of Proof of Work Blockchains. Tech. rep., IACR: Cryptology ePrint Archive, 2016.

[31] GERVAIS, A., RITZDORF, H., KARAME, G. O., AND CAPKUN, S. Tampering with the Delivery of Blocks and Transactions in Bitcoin. In *22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), ACM, pp. 692–705.

[32] GUERRAOUI, R., KNEŽEVIĆ, N., QUÉMA, V., AND VUKOLIĆ, M. The next 700 BFT protocols. In *5th European conference on Computer systems* (2010), ACM, pp. 363–376.

[33] HEARN, M., AND SPILMAN, J. Rapidly-adjusted (micro)payments to a pre-determined party, 2015.

[34] HEILMAN, E., KENDLER, A., ZOHAR, A., AND GOLDBERG, S. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In *24th USENIX Security Symposium* (2015), pp. 129–144.

[35] KARAME, G. O., ANDROULAKI, E., AND CAPKUN, S. Double-spending fast payments in Bitcoin. In *19th ACM Conference on Computer and communications security* (2012), ACM, pp. 906–917.

[36] KIAYIAS, A., AND PANAGIOTAKOS, G. Speed-Security Tradeoffs in Blockchain Protocols. Tech. rep., IACR: Cryptology ePrint Archive, 2015.

[37] KING, S., AND NADAL, S. PPCoin: Peer-to-peer Crypto-Currency with Proof-of-Stake.

[38] KOTLA, R., ALVISI, L., DAHLIN, M., CLEMENT, A., AND WONG, E. Zyzzyva: Speculative Byzantine Fault Tolerance. In *21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP)* (Oct. 2007), ACM.

[39] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.

[40] LENSTRA, A. K., AND WESOLOWSKI, B. A random zoo: sloth, unicorn, and trx. IACR eprint archive, Apr. 2015.

[41] LEWENBERG, Y., SOMPOLINSKY, Y., AND ZOHAR, A. Inclusive Block Chain Protocols. In *Financial Cryptography and Data Security*. Springer, Jan. 2015, pp. 528–547.

[42] LINUX FOUNDATION. Hyperledger Project, 2016.

[43] MAZIÈRES, D. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus.

[44] MERKLE, R. C. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Stanford University, June 1979.

[45] MILLER, A., AND JANSEN, R. Shadow-Bitcoin: scalable simulation via direct execution of multi-threaded applications. In *8th Workshop on Cyber Security Experimentation and Test (CSET 15)* (2015).

[46] MILLER, A., XIA, Y., CROMAN, K., SHI, E., AND SONG, D. The honey badger of BFT protocols. Tech. rep., Cryptology ePrint Archive 2016/199, 2016.

[47] NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[48] NAYAK, K., KUMAR, S., MILLER, A., AND SHI, E. Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. In *1st IEEE European Symposium on Security and Privacy* (Mar. 2015).

[49] PEASE, M., SHOSTAK, R., AND LAMPORT, L. Reaching agreement in the presence of faults. *Journal of the ACM (JACM) 27*, 2 (1980), 228–234.

[50] POON, J., AND DRYJA, T. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Jan. 2016.

[51] SCHNORR, C. P. Efficient signature generation by smart cards. *Journal of Cryptology 4*, 3 (1991), 161–174.

[52] SCHWARTZ, D., YOUNGS, N., AND BRITTO, A. The Ripple protocol consensus algorithm. *Ripple Labs Inc White Paper* (2014), 5.

[53] SOMPOLINSKY, Y., AND ZOHAR, A. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains, Dec. 2013.

[54] SYTA, E., TAMAS, I., VISHER, D., WOLINSKY, D. I., L., GAILLY, N., KHOFFI, I., AND FORD, B. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *37th IEEE Symposium on Security and Privacy* (May 2016).

[55] VENKATARAMAN, V., YOSHIDA, K., AND FRANCIS, P. Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast. In *14th International Conference on Network Protocols (ICNP)* (Nov. 2006).

[56] VUKOLIĆ, M. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security* (2015), Springer, pp. 112–125.

[57] WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* (2014).

[58] YU, H., GIBBONS, P. B., KAMINSKY, M., AND XIAO, F. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *29th IEEE Symposium on Security and Privacy (S&P)* (May 2008).