arXiv:1602.06997v1 [cs.CR] 22 Feb 2016

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 – 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 representing 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. Thanks to these optimizations ByzCoin achieves a throughput higher than Paypal currently handles, with confirmation latencies of 15-20 seconds.

1 Introduction

Bitcoin [40] is a decentralized cryptocurrency providing an open, self-regulating alternative to classical 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. Further, later work revealed additional vulnerabilities to transaction reversibility, double-spending, and strategic mining attacks [21, 25, 29, 30, 41].

The key problem is that Bitcoin's consensus algorithm provides only eventual consistency rather than strong consistency. Strong consistency could offer cryptocurrencies three important benefits. First, all miners agree on the validity of blocks right away, without wasting computational power resolving inconsistencies (*forks*). Second, clients need not wait extended periods for certainty 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. While strong consistency for cryptocurrencies has been suggested before [14, 15, 36, 46], existing proposals give up Bitcoin's decentralization, 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) [11] algorithm. ByzCoin addresses four key challenges in bringing PBFT-based strong consistency to cryptocurrencies: (1) open membership, (2) scalability to hundreds of replicas, (3) proof-of-work block conflicts, and (4) transaction commit rate.

PBFT was not designed for scalability to large consensus groups: deployments and experiments often employ the minimum of four replicas [32], and generally have not explored scalability levels beyond 7 [11] or 16 replicas [13, 27, 1]. ByzCoin builds PBFT atop CoSi [48], a collective signing protocol that efficiently aggregates hundreds or thousands of signatures. Collective signing reduces both the costs of PBFT rounds themselves and the costs for "light" clients to verify transaction commitment. Although CoSi is not itself a consensus protocol, ByzCoin builds a BFT protocol using two CoSi signing rounds for PBFT's *prepare* and *commit* phases.

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 [19]. ByzCoin addresses this conflict by forming consensus groups dynamically from *windows* of recently mined blocks, giving each recent miner *shares* or voting power proportional to their recent commitment of hash power.

Finally, to reduce transaction processing latency we adopt the idea from Bitcoin-NG [20] 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 and collectively sign blocks of Bitcoin's current maximum size (1MB) in less than 20 seconds. 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 with both number of participants and block size. For the one-day consensus group (144 miners), 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. We find that ByzCoin enables clients to confirm transactions, without risk of reversal or double-spending, within 30 seconds of issuance. Assuming an attacker controlling less than 33% of the computing power, Bitcoin requires more than 4 hours to reach comparable confidence of irreversible commitment.

ByzCoin is still an early proof-of-concept with important limitations. First, ByzCoin does not improve on Bitcoin's proof-of-work mechanism; we consider finding a suitable replacement [3, 23, 31, 51] an important but orthogonal issue for future work. Like many BFT protocols in practice [12, 27], ByzCoin may be vulnerable to slowdown or temporary DoS attacks that Byzantine nodes can trigger. Further, while 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 their payments, and/or attempt to censor transactions. PBFT is secure only against attackers controlling 34% of hash power, not 51% – though Bitcoin has analogous weaknesses accounting for selfish mining [21]. Finally, ByzCoin's security is at present analyzed only informally (Section 5).

This paper makes the following key contributions:

- We present the first demonstrably practical Byzantine consensus protocol for decentralized Bitcoin-like cryptocurrencies based on proof-of-work.
- We use collective signing [48] to scale BFT protocols to large consensus groups and enable clients to very transaction commitment efficiently.
- We demonstrate experimentally that a strongly-consistent cryptocurrency can increase Bitcoin's throughput by two orders of magnitude with transaction confirmation latencies under one minute.

The remainder of the paper is organized as follows. Section 2 summarizes core concepts of Bitcoin and its variations, scalable collective signing, and Byzantine fault tolerance. Section 3 details the ByzCoin protocol. Section 4 describes an evaluation of our prototype implementation of ByzCoin. Section 5 informally analyzes ByzCoin's security and Section 6 outlines related work,

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 [40] 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 such that new blocks are mined about every ten minutes on average. Each block includes a Merkle tree [37] 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 Bitcoin network, who (when behaving properly) accepts the new block if it represents a valid chain strictly longer than any they have already seen.

Bitcoin's decentralized consensus and security derives 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, follow these rules and always attempt 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. Its security is guaranteed by the fact that this majority will be extending the legitimate chain faster than any corrupt minority that might wish to rewrite history or double-spend currency. Bitcoin's consistency guarantee is only probabilistic and eventual, however, leading to two fundamental problems.

First, multiple miners may 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 gets extended yet again. Subsequently, all well-behaved miners on the shorter chain(s) switch to the new longest one. All transactions in the rejected block(s) are then invalid and have to be resubmitted for inclusion into the winning blockchain. Thus, Bitcoin clients desiring high probabilistic 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, increasing the time interval until a transaction can be considered completed. As a rule of thumb, 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 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 also increases the probability of forks due to higher propagation delays and the risk of double-spending attacks [47]. 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 [25, 14].

Bitcoin-NG. Bitcoin-NG [20] 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. Leveraging 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 allows Bitcoin-NG to process many transaction microblocks between the mining of two keyblocks, allowing transaction throughput to increase.

Bitcoin-NG retains many drawbacks of Bitcoin's eventual consistency model, however. 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 or more minutes. 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. While Bitcoin-NG includes disincentives for such behavior, these disincentives amount at most to the "mining value" of the keyblock (coinbase rewards and transaction fees): leaders are thus both able and incentivized to double-spend on higher-value transactions.

Thus, while 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 we build on in Section 3.5 to support high-throughput, lower-latency transactions with strong consistency in ByzCoin.

2.2 Byzantine Fault Tolerance

The *Byzantine Generals' Problem* [33, 42] 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. [42] 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 [11] was the first efficient solution to the Byzantine Generals' Problem and works in eventually synchronous environments such as the Internet. PBFT offers both *safety* and *liveness* provided that the above bound is adhered, 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, 13, 32, 27].

Every round of PBFT has 3 distinct phases. First is the pre-prepare phase where the primary of the view announces the next record that the system should agree upon. On receiving the 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 $(2f+1)$ of prepare messages and publish this observation with a commit message. Then they wait for a quorum $(2f+1)$ of commit messages to make sure that enough nodes are ready to apply any decision. If consequent rounds of PBFT finish out of order, the changes are stashed and implemented in order.

As it can be seen, the system relies upon a correct primary to start each round and proceeds if a quorum exist, consequently the leader is an attack target. For this reason PBFT has a view change protocol that ensures liveness with a faulty leader. All nodes monitor the leader's actions and if they detect any malicious behaviour or failure, initiate a view-change. They announce their decision to change leader and stop validating the leader's actions. If a quorum $(2f+1)$ of nodes decides that the leader is faulty then the leader that was indicated in the view change message takes over.

PBFT also has limitations. First, it assumes a fixed,

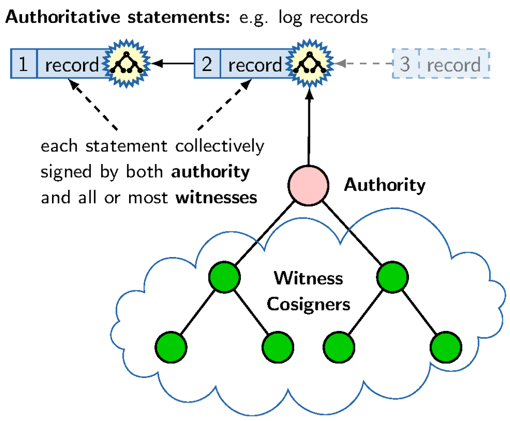

Figure 1: CoSi protocol architecture

well-defined group of replicas, contradicting one of Bitcoin's basic principles 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 might represent hundreds or thousands of Bitcoin nodes. Third, after submitting a transaction to a PBFT service, a client must communicate with a supermajority of the replicas in order to confirm the transaction has committed and learn its outcome, making secure transaction *verification* unscalable.

2.3 Scalable Collective Signing

CoSi [48] 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 run of the protocol yields a single digital signature with size and verification cost comparable to an individual signature, but compactly attests that both the leader and perhaps many witnesses observed and agreed to sign the statement.

To achieve scalability, CoSi combines Schnorr multi-signatures [45] with communication trees of the type long used in multicast protocols [10, 17, 49]. 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 requiring 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 it 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 commit from its immediate children in the communication tree, 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:** Based on the collective challenge, all nodes compute an aggregate response in bottom-up fashion mirroring the commitment phase.

The result of this four-phase protocol is to produce a largely standard Schnorr signature, requiring about 64 bytes in size using the Ed25519 [5] that anyone can verify against the aggregate public key 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 may 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 [48].

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 Strawman Design: PBFTCoin

For simplicity we start with PBFTCoin, an unrealistically simple protocol combining PBFT with Bitcoin in a naive fashion, 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 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 can never be rolled back or rewritten. Prior work has suggested essentially such a design [14, 15], though without addressing the scalability challenges it creates.

Under these simplifying assumptions PBFTCoin guarantees safety and liveness, since at most f nodes are faulty and thus the usual BFT security bounds apply. However, the assumption of having a postulated, fixed group of trustees is unrealistic for Bitcoin-like decentralized cryptocurrencies allowing open membership. Moreover, since PBFT trustees authenticate each other via non-transferrable symmetric-key MACs, each trustee must communicate directly with most other trustees in every round, 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 reduce the total number of messages required from $O(n^2)$ to $O(n)$.
3. We introduce scalable collective signing to further reduce per-round communication complexity 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.

3.2 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 [19] 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 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 the share window is moved one step forward. Old shares beyond the current window effectively expire and become useless for purposes of consensus group membership. Miners holding no more valid shares in the current window loose their membership in the consensus group and thus are no longer allowed to participate in the decision making.

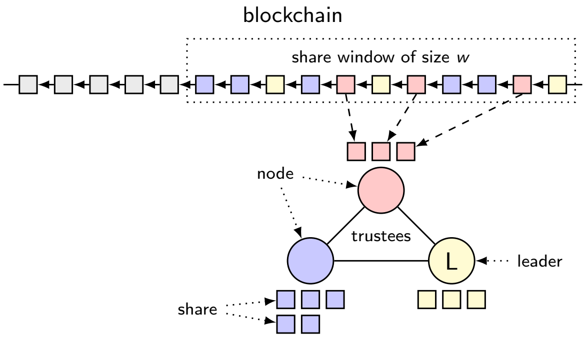

Figure 2: Valid shares for mined blocks in the blockchain are credited to miners

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 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 (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 of time and threatening the balance in the consensus group. The relationship between blocks, miners and shares is illustrated in Fig. 2.

We leave to future work a detailed exploration of how large the window should be in practice. In general, shorter windows produce smaller consensus groups, mitigating BFT scalability challenges, but produce coarser "samples" of recent hash power over a shorter period, potentially leaving the system vulnerable to "flash" attackers who might invest a great deal of hash power over a short time period. Longer windows, on the other hand, produce more accurate samples of hash power over longer periods, but could create an increased risk to the system's liveness if there is too much "churn" in mining power over the course of the window: *e.g.*, if miners representing more than one-third of in-window blocks suddenly leave the network before their shares have expired.

Mining Rewards and Transaction Fees. Since we can no longer assume voluntary participation as in PBFT-Coin'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 existing 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, and hence the more shares the miner holds, the higher the revenue the miner receives during payouts in the current window. This division of rewards also incentivizes consensus group members to remain live and participating, since 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.

One problem is that the leader might censor transactions, since in our current protocol the leader controls which blocks are proposed and the set of transactions they contain. While some similar censorship risks already exist in Bitcoin and we defer a full solution to future work, we mitigate this risk in part via PBFT's standard view-change protocol. Any node who detects that the leader is dishonestly ignoring or omitting transactions from live nodes, holds no more shares, or has failed in other detectable ways, broadcasts a view change message and stops accepting blocks from the current leader. When $2f+1$ nodes have sent a view-change message, the previous leader becomes incapable of processing further transactions and the next leader takes over.

A second limitation of the current protocol is that a malicious leader might choose to send a message to only slightly over $\frac{2}{3}$ of current consensus group members and still make progress, depriving rewards from up to $\frac{1}{3}$ of members who might in fact be online but "out of favor" with the leader. While we again defer a full solution to future work, our current design at least removes financial incentives for such strategies by specifying that the reward-portions of purportedly "offline" trustees are not redistributed to the online ones but are merely discarded. As with the transaction censorship problem, honest nodes could also in principle gossip with each other to detect a leader's malicious inclusion of live members, and initiate a view-change in response.

Alternatives to Hash Power. 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 [3], or proof-of-stake designs [31]. Consensus group membership might in principle also be based on other Sybil attack-resistant methods, such as those based on social trust networks [51]. 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 [23]. Regardless, we treat the ideal choice of Sybil attack-resistance mechanism as an issue for future work orthogonal to the focus of this paper.

3.3 Replacing MACs by Digital Signatures

In our next refinement step towards ByzCoin, we tackle the scalability challenge caused by PBFT's non-transferrable MAC-based message authentication. Since MACs rely on pairwise shared secrets, all trustees must exchange messages pairwise in each round to reach consensus, resulting in a communication complexity of $O(n^2)$ where n is the group size. While this works for small groups it does not scale.

PBFT's decision on point-to-point communication patterns was motivated by the desire to avoid public-key operations on the critical transaction path. However, the cost for public-key operations has decreased thanks to well-optimized asymmetric cryptosystems [5], making those costs less of an issue. By adopting digital signatures for authentication, we are able to use sparser and more scalable communication topologies enabling the current leader to collect and distribute third-party verifiable evidence that certain steps in PBFT have succeeded. This removes the necessity that all trustees communicate directly with each other. With this measure we can use tree-based communication structures as done in well-known multicast protocols [10, 17, 49], reducing the total number of messages from $O(n^2)$ to $O(n)$.

3.4 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 may be practical. If we desire 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 [48] for collective signing. CoSi does not directly implement consensus or BFT, but 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 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 this first CoSi round generates implements the PBFT protocol's *prepare* phase, in which the leader obtains attestations from a two-thirds supermajority quorum of consensus 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 is insufficient by itself to ensure that the proposal *will be* committed. The leader and/or some number of other members could fail before a supermajority 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 supermajority that all the signing members witnessed the successful result of the prepare phase and make a positive commitment to remember the decision. This collective signature resulting from this second CoSi round effectively attests that a two-thirds supermajority of members not only considers the leader's proposal "safe" but promises to remember it, and hence that the leader's proposal has 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 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 will 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 supermajority 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 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, however, 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 may verify in $O(1)$ computation time, that a given block – and hence any transaction within that block – has irreversibly committed. This secondary scalability benefit may in practice be more important than the first, because it allows "light clients" that 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.

3.5 Decoupling Transaction Verification from Leader Election

While ByzCoin so far provides a scalable guarantee of strong consistency, 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 may nevertheless be significant: *e.g.*, up to 10 minutes using Bitcoin's mining difficulty tuning scheme. While 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, just as in Bitcoin [40]. To enable lower client-perceived transaction latencies, therefore, we build on the important idea of Bicoin-NG [20] 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 3.2. As in Bitcoin-NG, this decoupling allows the current leader to propose and commit many microblocks, containing 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

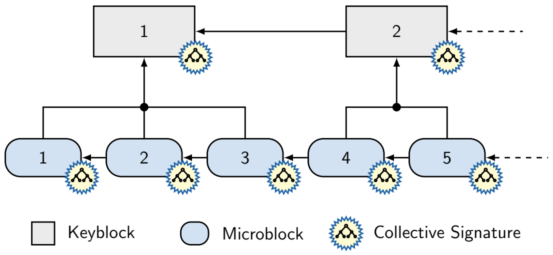

Figure 3: ByzCoin blockchain: Two parallel chains store information about the leaders (keyblocks) and the transactions (microblocks)

as shown in Fig. 3. The main blockchain is the keyblock blockchain, consisting of all mined blocks. The microblock blockchain is a secondary blockchain which 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. The microblock is a simple block that is validated every few seconds. It includes the transactions that it verifies. There is also the aggregated public key and the signature. It refers to the previous microblock and keyblock (microblock in order to ensure total ordering and keyblock to indicate who is its creators and signers). Its hash is collectively signed by the CoSi tree of the corresponding round.

Keyblocks. The keyblock is used to determine consensus group membership and to pay signers their rewards, since adding N transactions inside every microblock would cause bigger blocks and slower propagation. In the keyblock the miner should include the fee payments of the round before the previous one, so that the keyblock contents are known before the mining process begins.

3.5.1 Leader Election Issues

Decoupling transaction verification from the block-mining process comes with a cost. Until now we assumed that the leader of the system is 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 leader election purpose. To resolve this issue we define a mandatory view-change every time a keyblock is signed, to the keyblock's miner. That way the power of veryfing 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.

Tree Creation in ByzCoin. Once a miner A mines a keyblock, its first action would be to validate it via CoSi. To do that, A needs to form a communication tree with itself as the leader. Currently, in order for the leader to change in CoSi, all the miners need to acknowledge it individually and decide on the next view, which requires $O(N)$ communication steps. To avoid this overhead at the beginning of a round, during the previous round the miners autonomously create the next tree bottom-up. 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, and 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, which 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 $i-1$.

3.6 Block Conflicts and Selfish Mining

Strong consistency by definition does not allow inconsistencies like forks. However the way the miners collectively decide how to resolve block conflicts can still allow selfish mining [21] to happen like in Bitcoin. Worse, if the miners decide randomly to vote for one of the two blocks, there is the possibility that no group reaches the super-majority that BFT needs, which allows the previous leader to keep making microblocks instead of handing this power over to the next leader. Another approach to voting would be to apply a deterministic function on the blocks. Many solutions can be selected (lowest sum of bits wins, hashing and the smallest hash wins etc.), but they can all be reduced to the smallest hash wins problem, which is native in Bitcoin since the proof-of-work system needs nothing more than an adequately small hash. Unfortunately, this practice could encourage selfish mining as explained in Appendix A.

One way to tie-break without helping selfish miners, as proposed in [21], is to increase the entropy of the output of this deterministic process. We follow this idea using the following algorithm that is run by all miners at the same time. On a fork all the blocks' header hashes are put in an array so that the array remains sorted (low to high). The array is then hashed and the final bit(s) determine the index showing which block wins the fork.

The solution, in Fig. 4, also uses the idea of a deterministic function applied to the blocks, requiring no voting. Its advantage is that the input of the hash function is partially unknown before the fork occurs, thus the en-

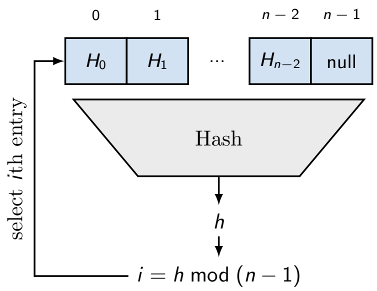

Figure 4: Deterministic fork resolution in ByzCoin

tropy of the output is high and difficult for an attacker to be able to optimize. Given that the search space of a possible conflict is as big as the search space of 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.

3.7 Tolerating Churn and Byzantine Faults

This section discusses the challenges of fault tolerance in ByzCoin, particularly concerning tree failures and the maximum number of tolerable Byzantine faults.

3.7.1 Tree Fault Tolerance

In CoSi there are two different mechanisms that allow substantial fault-tolerance. Furthermore the strict membership requirements and the frequent tree changes of ByzCoin increase the difficulty of a malicious attacker with less than 33% of the total computing power to attack the system. However, a security analysis 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 [27], modifying ByzCoin so that the tree communication pattern is a normal-case performance optimization that can withstand most malicious attacks. However, when the liveness of the tree-based ByzCoin is compromised the leader can decide to get back to the 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 which 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 gets $\frac{2}{3}$ of the ACKs, the leader knows that the tree has failed and reverts to a flat ByzCoin before the witnesses assume that he is faulty. As we show in Section 4, the flat ByzCoin can still quickly sign keyblocks for the day-long grid (144 witnesses) and can maintain a throughput higher than the one Bitcoin currently has.

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 epoch 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 membership, to sign a latest checkpoint message containing the hash of the last microblock. At that point of 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 Implementation and Evaluation

This section describes the current prototype implementation of ByzCoin and discusses various experiments.

4.1 Prototype Implementation

We have built a working prototype of ByzCoin written in Go [26]. The prototype implements both flat and tree-based collective block signing as described above. We also implemented a conventional PBFT consensus algorithm with the same communication framework for comparison, and a tree-based consensus algorithm using individual signatures. We evaluated ByzCoin with Schnorr signatures implemented with the Ed25519 curve [5]. The prototype is open source and publicly available *(link omitted for anonymous review)*.

ByzCoin is implemented on top of the CoSi signing protocol. There are also clients parsing real Bitcoin

		t_i	t_{i+1}	t_{i+2}	t_{i+3}	t_{i+4}
B_k	prepare	An/Co	Ch/Re			
	commit		An/Co	Ch/Re		
B_{k+1}	prepare		An/Co	Ch/Re		
	commit			An/Co	Ch/Re	
B_{k+2}	prepare			An/Co	Ch/Re	
	commit				An/Co	Ch/Re

Table 1: ByzCoin pipelining for maximum transaction-throughput; B_k denotes the microblock signed in round k, An/Co the Announce-Commit and Ch/Re the Challenge-Response round-trips of CoSi

transactions and sending them to the servers to create blocks. Both microblocks and keyblocks are fully transmitted and collectively signed through the tree and are returned, together with the proof, to the clients upon request. The verification of the block header is implemented but transaction verification within blocks is only emulated due to hardware scarcity. The same practice is proposed in Shadow Bitcoin [38]. We base our emulation both on measurements [25] of the average block verification delays (around 200 ms for 500 MB blocks) and the claims of Bitcoin developers [7] that as far as hardware is concerned Bitcoin can easily verify 4000 TPS. We simulate a linear increase of this delay based on 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.

CoSi Pipelining. Collective signing [48] is done in four different phases per round, namely announce, response, challenge, and commit. In ByzCoin the announce and commit phases of CoSi can be performed in advance before the block to be committed is available, since the proposed block may be sent to the signers in the challenge phase. This enables us to pipeline two rounds so that the announce/commit phases of ByzCoin's commit round are piggybacked on the challenge and response messages of the prepare round. This pipeline reduces latency by one round-trip over the communication tree. Looking into the normal execution of ByzCoin, this pipeline can be extended so that an optimal throughput of one signed microblock per round-trip is produced. A sample execution can be seen in Table 1.

4.2 Evaluation

The main question we want to evaluate is whether ByzCoin is usable in practice without incurring large overhead. We evaluated keyblock and microblock signing for an increasing number of consensus group members and ByzCoin latency for an increasing microblock size, for all implemented consensus algorithms. We then compare Bitcoin with the flat and tree-based versions of ByzCoin to investigate the maximum throughput that each variant can achieve. In another experiment we investigate the latency of signing single transactions and larger blocks. Finally, we demonstrate the practicality of ByzCoin as far as latency for securing transactions is concerned, from a client's point of view.

4.2.1 Experimental Setup

We evaluated the prototype on DeterLab [18], using 36 physical machines configured in a star-shaped virtual topology. To simulate larger numbers of ByzCoin witnesses than available test-bed machines, we run up to 28 separate ByzCoin processes on each physical machine. We also run a client process to generate the necessary transactions for filling the blocks. To mimic a realistic wide-area network environment we imposed a round-trip latency of 200 ms between any two machines and a link bandwidth of 35 Mbps per simulated host.

4.2.2 Scaling With Consensus Group Size

This experiment focuses on the scalability of ByzCoin's BFT protocol in terms of number of consensus group members. The number of unique miners participating in a consensus group is limited by the number of membership shares in the window (Section 3.2), but may be smaller if some miners hold multiple shares (*i.e.*, successfully mined several blocks) within the same window.

We have run 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. 5 shows, the simple version of ByzCoin achieves acceptable latency as long as 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 other hand the tree-based ByzCoin scales well, since the same 1 MB block suffers less signing latency for 1008 nodes than the flat approach for 36 nodes. Adding 28 times more nodes (from 36 to 1008) imposes a latency increase close to a factor 2 (from 6.5

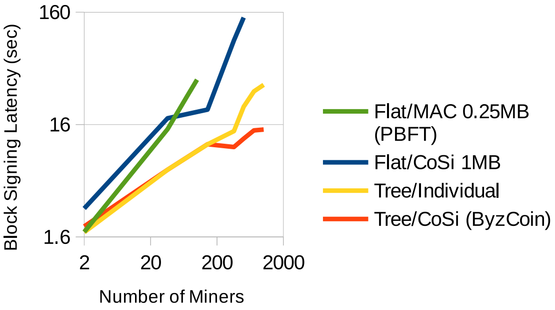

Figure 5: ByzCoin consensus latency versus number of participating nodes

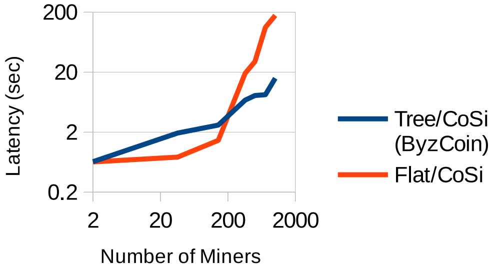

Figure 6: Keyblock signing latency

to 14 seconds). The basic PBFT implementation is quite fast for 2 nodes but scales poorly (40 seconds for 100 nodes), while the tree-based implementation with individual signatures performs the same as ByzCoin until 200 hosts. If we want the higher security of 1008 nodes, however, then ByzCoin is 3 times faster.

Fig. 6 shows the performance cost of keyblock signing. The reason the flat variant seems to outperform the tree when the number of hosts is small is that the blocks are also small (they have as many transactions as hosts). This leads to fast transmission even in the flat case, so the overhead comes from the propagation latency, which scales with $O(\log N)$ in the tree-based ByzCoin variant.

4.2.3 Scaling With Block Size

The next experiment focuses on the effects of block size on the scalability of the system. We tested variable block sizes for a constant number of hosts. We tested the proposed 144 node system for all implementations. Once again PBFT was unable to achieve acceptable latencies with 144 nodes, thus we ran it with 100 nodes.

Fig. 7 shows average latency out of 10 blocks as microblock size increases. As in the previous section we can see that the flat implementation is acceptable for a 1 MB block, but when the block increases to 2 MB the latency quadruples. This is expected since the leader's link saturates when he tries to send 2 MB messages to every participating node. In contrast ByzCoin scales well since the leader outsources the transmission of the blocks to other nodes and only contacts his children. The same behavior is observed for the individual signatures tree algorithm, which shows that block size has no negative effect in 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 250KB blocks.

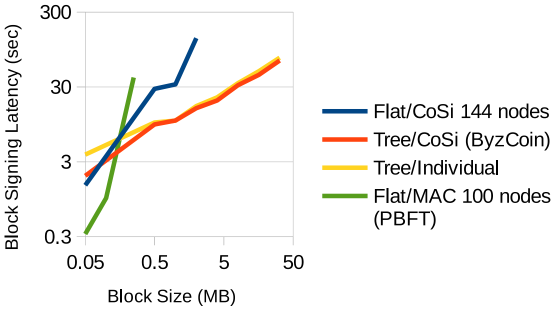

Figure 7: ByzCoin consensus latency versus block size

ByzCoin's signing latency for a block size of 1 MB is close to 10 seconds of delay, small enough to make the need for 0-confirmation transactions disappear. Even for a 32 MB block (around 66000 transactions) the delay remains much smaller than Bitcoin's ≈ 10 minutes.

4.2.4 Scaling of Various Block Sizes

This experiment is based on the need to defend against 0-confirmation attacks even if the transaction needs to be completed in less than 10 seconds. We propose that the ByzCoin consensus group can offer another service of collectively signing transactions. These transactions are not directly added to the blockchain, but if signed the miners guarantee to add them into the block after the next. Adding them in the next block will interrupt and alter the mining process and also introduces race conditions, since a transaction can be signed while the block is signed. This is also a corner-case where only f fault out of $3f+2$ nodes can be tolerated.

Fig. 8 demonstrates the signing latencies of various blocks sizes. 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, the 1 MB and 2 MB at first scale linearly in number of nodes, but after 200 nodes the propagation latency is higher than the transmission of the block and thus the latency is close to constant. For 8 MB

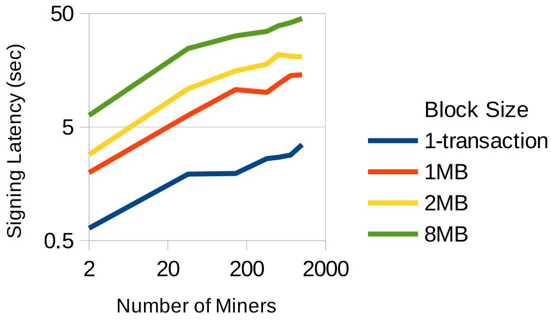

Figure 8: Block signing latencies versus number of hosts

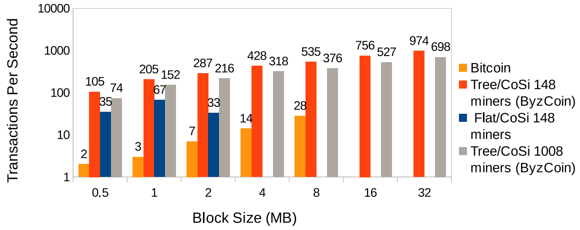

Figure 9: Throughput of ByzCoin

blocks, even with 1008 the signing latency increases linearly.

4.2.5 Throughput

In this experiment we investigate the maximum TPS that ByzCoin can achieve. We tested both versions with 144 and 1008, respectively. We also show how the flat version of ByzCoin performs and what throughput Bitcoin could get with varying block sizes.

In Fig. 9 we see that Bitcoin can increase its throughput by more than one order of magnitude by separating the transaction verification from the Block mining and dealing with forks via strong consistency (flat-ByzCoin). Furthermore the 144 grid can achieve close to 1000 TPS which is double the throughput of Paypal, and even the 1008 grid is close to 700 TPS. Finally, even when the tree fails the system can revert back to 1 MB microblock on the flat and more robust variant of ByzCoin and still have a throughput ten times higher than the current maximum throughput of Bitcoin.

Looking at both Figure Figs. 7 and 9, the usual tradeoff between throughput and latency appears. The system can work when the load is normal with 1–2 MB blocks where the latency is between 10–20 seconds and, when an overload happens, the system adaptively changes the block size to allow for higher throughput. We note that this is not the case in the simple ByzCoin where 1 MB blocks had optimal throughput and acceptable latency.

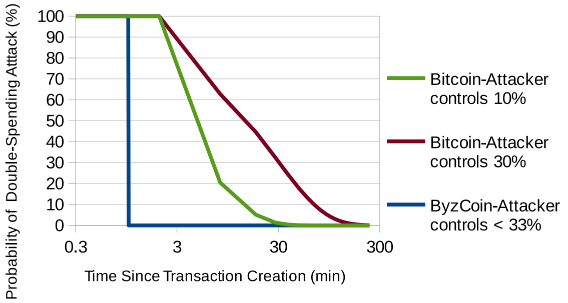

Figure 10: Successful double-spending attack probability

4.2.6 Transaction Safety Latency of ByzCoin

In the original Bitcoin [40] paper, Nakamoto models the security of a transaction when an attacker tries to double spend it, as a Gambler's Ruin Problem. Furthermore, he models the progress an attacker can make as a Poison distribution and combines these two models to reach equation Eq. (1). This equation calculates the probability of a successful double spend after z blocks when the adversary controls q computing power.

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

In Figs. 10 and 11 we used data from `blockchain.info` [8] to calculate the relative safety of a transaction through time. In Fig. 10 it can be seen that ByzCoin can secure a transaction in less than a minute, due to the collective signature. On the contrary Bitcoin's transactions need hours to be considered completely secured from a double-spending attempt.

Furthermore Fig. 11 illustrates the time needed to elapse from the transaction creation until there is 99.9% security, *i.e.*, until a double-spending attack has less than a 0.01% chance of success. ByzCoin achieves the same sub-minute latency in the time the system needs to produce a collectively signed microblock, while Bitcoin requires several hours. This graph does not consider other advanced attacks like eclipse attacks [29], where Bitcoin offers no security for the victim's transactions.

5 Security Analysis

In this section we analyze the most common Bitcoin attacks and how ByzCoin's consensus mechanism mitigates or eliminates them.

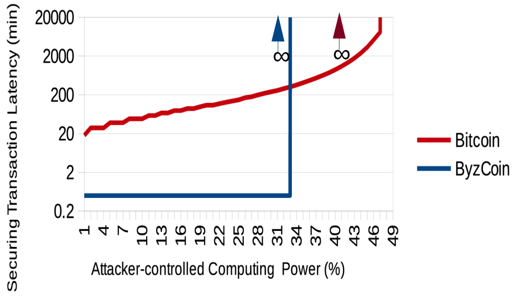

Figure 11: Client-perceived secure transaction latency

5.1 Defense Against Bitcoin Attacks

Race and Finney Attacks. 0-confirmation double-spends is a family of attacks that happen when real-time service is needed and the merchant cannot afford to make the client wait for 10 or more minutes. In this case the merchant releases the goods after sending the transaction to the network. If the attacker at the same point creates a conflicting transaction transferring the funds to him and publishes it to the network, then this is a race attack [30]. If the client has already mined a block including the double-spent transaction then this is a Finney attack [22]. ByzCoin can mitigate both attacks by putting the merchant's transaction in a collectively signed microblock whose verification latency is a few seconds. That way if the merchant can wait these few seconds to one minute, then the risk of a double spending attack is eliminated. However, if this latency is also unacceptable then he can send a single transaction for signing, which will cost more fees, but secures the transaction in less than 3 seconds.

N-Confirmation Double-spend Attack. In the N-confirmation attack [6], the attacker sends a transaction to a merchant who waits for its inclusion in a block and concludes the interaction after $N-1$ blocks. After that the attacker creates a new double-spending transaction and tries to fork the blockchain. If the attacker has enough computing power there is a real possibility that he can do it. For example [40] for $N = 3$, if he has 10% of the computing power he has 5% chances of succeeding .

In ByzCoin the merchant will know that his transaction is accepted in a block that is widely accepted as correct by verifying the collective signature. Afterwards the attacker cannot succeed in forking the blockchain because 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 happen.

Eclipse and Delivery Tampering Attacks. Eclipse attack has been introduced in [29]. The adversary eclipses the victim by controlling all its connections with the Bitcoin network and thus all the information it gets. This way the attacker can mount both 0-confirmation and N-confirmation attacks with a higher chance of succeeding and after longer time periods.

Another attack is introduced by Gervais et al. [25] who show how to exploit the scalability measures that Bitcoin uses to delay the delivery of new blocks. That way the attacker can control the information that the victim receives and thus prevents the latter from updating its view on the blockchain. That way an adversary can easily double-spend 0- and 1-confirmation transactions. Moreover, selfish mining is simplified. As demonstrated above, the victim can be protected from both attacks in ByzCoin thanks to collective signing.

Selfish and Stubborn Mining Attacks. Finally Eyal et al. [21] show that any miner can gain more profit by trying to build a hidden blockchain every time it mines a block, especially if the miner has a good connection to the Bitcoin network. The authors propose a counter measure that makes this attack impossible for miners with less than 25% computing power (33% if the network was optimal).

An extension of this attack is the so-called stubborn mining [41, 44] which generalizes the idea behind selfish mining to increase the adversaries revenue. These strategies revolve around the decisions an attacker should follow in order to overtake the public blockchain with his private one and combine it with eclipse attack [41, 29].

In ByzCoin these strategies are of no use since forks are instantly resolved in a deterministic manner, thus building a hidden blockchain is only wasting resources and minimizing revenue.

5.2 Drawbacks and Limitations

We have already mentioned some drawbacks of ByzCoin in Section 3.2, but there is one more drawback directly related to security. An attacker with more than 33% of the computing power can double spend, by getting two conflicting blocks signed. In Bitcoin double-spending with less than 50% of the computing power is possible even with low computing power but not as certain as it is in ByzCoin with 33% of the computing power. This can be seen in **??** where safety of ByzCoin fails at 33% while Bitcoin remains safe if a client can wait long enough even for 48%.

6 Related Work

ByzCoin has the same goals as Bitcoin [40] and the related efforts to improve it [50, 47, 34]; the implementation of a State Machine Replication system where membership is open. The same assumptions are explored in multiple research papers [39, 9, 24]. Our model defers from classic Byzantine fault-tolerant SMRs, by not assuming static or slowly changing membership [11, 35].

As far as cryptocurrencies are concerned, there is a big debate about increasing the block size, hence the maximum throughput in Bitcoin. This increase does not eliminate the inherent limitation of Bitcoin, which stems from the eventual consistency where forks occur. A first notable effort for scalability was the GHOST protocol [47] that changes the chain selection rule. In Bitcoin, the chain that is selected as valid is the one that includes the most work, whereas Ghost suggests that the sub-tree that includes the most work should be selected at a fork. This way the sub-tree that was considered correct the longest time will have high possibility of being selected, making an intentional fork much harder. One of the biggest limitation of GHOST is that in actual operation time no nodes will know the full tree, since invalid blocks are not propagated. One solution is to propagate all the blocks, but this opens the door for an attacker to flood the network with low-difficulty blocks.

An alternative to improving the latency and throughput of the Bitcoin network is to allow off-chain transactions. This idea draws its origin from Hearn and Spilmans two-point channel protocol [28]. The Lightning Network [43] and the micro-payment channels [16] enable payment networks where transactions occur without a trusted middleman. They use contracts to allow any party to generate proof of fraud on the main blockchain and to deny revenue from an attacker. Although these systems enable a faster cryptocurrency, they do not address the problem of scaling a SMR system, thus sacrifice the open and distributed nature of Bitcoin. Finally, the use of side chains is proposed, where transactions can move bitcoins from one chain to another [4]. This enables the distribution of the workload to multiple sub-sets of nodes running the same protocol. This approach can be used in ByzCoin by having different ByzCoin trees for each side-chain

As far as BFT is concerned, Hyperledger [35] implement and run a PBFT variant of the blockchain, but the membership changes are slow, if existent at all and, its decentralization depends on the configuration. An interesting alternative proposed in Bitcoin meets Strong Consistency [15] shares similarities with our approach, but is only a preliminary theoretical analysis. Finally Stellar [36] proposes a novel consensus protocol named Federated Byzantine Agreement that introduces Quorum Slices enabling an ”open for anyone to participate” BFT protocol, but its security depends again on the right configuration of trustees by the client.

7 Conclusion

We have presented a new consensus algorithm for Bitcoin. It increases Bitcoin's core security guarantees, shields against attacks on the consensus and mining system, including N-confirmation double spending, intentional forks in the blockchain and selfish mining and enables high scalability and low latency of Bitcoin Transactions. Our system can be deployed to any blockchain-based system because the leader election mechanism of proof-of-work can easily change to another mechanism such as proof-of-stake or be static if openness is not an issue. We have developed a wide scale prototype implementation of our mechanism, and have validated its efficiency with measurements and experiments and showed that Bitcoin can increase the capacity of transactions it handles by more than two orders of magnitude. Finally we reasoned about the effectiveness of our strongly consistent consensus mechanism against most of the current Bitcoin attacks.

References

[1] Michael Abd-El-Malek, Gregory R. Ganger, Garth R. Goodson, Michael K. Reiter, and Jay J. Wylie. Fault-scalable byzantine fault-tolerant services. *SIGOPS Operating Systems Review*, 39(5):59–74, October 2005.

[2] Gavin Andresen. Classic? Unlimited? XT? Core?, January 2016.

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

[4] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. October 2014.

[5] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.

[6] Bitcoin Wiki. Confirmation, 2016.

[7] Bitcoin Wiki. Scalability, 2016.

[8] Blockchain.info. Median transaction confirmation time, February 2016.

[9] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua Kroll, and Edward W Felten. Research perspectives and challenges for Bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy. IEEE*, 2015.

[10] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[11] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, February 1999.

[12] Allen Clement, Edmund L Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti. Making Byzantine fault tolerant systems tolerate byzantine faults. In *6th USENIX Symposium on Networked Systems Design and Implementation*, April 2009.

[13] James Cowling, Daniel Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In *7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 177–190, Berkeley, CA, USA, 2006. USENIX Association.

[14] Kyle Croman, Christian Decke, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gun Sirer, Dawn Song an, and Roger Wattenhofer. On scaling decentralized blockchains (a position paper). In *3rd Workshop on Bitcoin and Blockchain Research*, 2016.

[15] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin Meets Strong Consistency. In *17th International Conference on Distributed Computing and Networking (ICDCN), Singapore*, January 2016.

[16] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18. Springer, August 2015.

[17] Stephen E. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2), May 1990.

[18] DeterLab network security testbed, September 2012.

[19] John R. Douceur. The Sybil attack. In *1st International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[20] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, Santa Clara, CA, March 2016. USENIX Association.

[21] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[22] Hal Finney. Best practice for fast transaction acceptance – how high is the risk?, February 2011. Bitcoin Forum comment.

[23] Bryan Ford and Jacob Strauss. An offline foundation for online accountable pseudonyms. In *1st International Workshop on Social Network Systems (SocialNets)*, 2008.

[24] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT 2015*, pages 281–310. Springer, 2015.

[25] Arthur Gervais, Hubert Ritzdorf, Ghassan O Karame, and Srdjan Capkun. Tampering with the delivery of blocks and transactions in Bitcoin. In *22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 692–705. ACM, 2015.

[26] The Go programming language, January 2015.

[27] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 BFT protocols. In *5th European conference on Computer systems*, pages 363–376. ACM, 2010.

[28] Mike Hearn and J Spilman. Rapidly-adjusted (micro)payments to a pre-determined party, 2015.

[29] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *24th USENIX Security Symposium*, pages 129–144, 2015.

[30] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in Bitcoin. In *19th ACM Conference on Computer and*

communications security, pages 906–917. ACM, 2012.

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

[32] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine fault tolerance. In *21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*. ACM, October 2007.

[33] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[34] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security*, pages 528–547. Springer, January 2015.

[35] Linux Foundation. Hyperledger Project, 2016.

[36] David Mazières. The Stellar consensus protocol: A federated model for Internet-level consensus. November 2015.

[37] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Stanford University, June 1979.

[38] Andrew Miller and Rob Jansen. Shadow-bitcoin: scalable simulation via direct execution of multi-threaded applications. In *8th Workshop on Cyber Security Experimentation and Test (CSET 15)*, 2015.

[39] Andrew Miller and Joseph J. LaViola Jr. Anonymous Byzantine consensus from moderately-hard puzzles: A model for Bitcoin. Technical Report CS-TR-14-01, University of Central Florida, April 2014.

[40] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, October 2008.

[41] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. Technical report, March 2015.

[42] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[43] Joseph Poon and Thaddeus Dryja. The Bitcoin lightning network: Scalable off-chain instant payments, January 2016.

[44] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in Bitcoin. In *Financial Cryptography and Data Security*, February 2016.

[45] Claus P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[46] David Schwartz, Noah Youngs, and Arthur Britto. The Ripple Protocol consensus algorithm, 2014. Ripple Labs White Paper.

[47] Yonatan Sompolinsky and Aviv Zohar. Accelerating Bitcoin's transaction processing. fast money grows on trees, not chains, December 2013.

[48] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Linus Gasser, Nicolas Gailly, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *37th IEEE Symposium on Security and Privacy*, May 2016.

[49] Vidhyashankar Venkataraman, Kaouru Yoshida, and Paul Francis. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *14th International Conference on Network Protocols (ICNP)*, November 2006.

[50] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2014. Ethereum Project Yellow Paper.

[51] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. In *29th IEEE Symposium on Security and Privacy (S&P)*, May 2008.

Appendix A : Selfish mining elimination in ByzCoin

If the smallest hash wins is used then a selfish miner can be lucky and get a smaller than needed hash of the block. For example, the target can be 10 leading zero bits but the miner finds a block with 12 leading zero bits ($\frac{1}{2^2}$ probability). Then this block has a 87,5% chance of winning a potential fork, thus it might be better to keep it hidden and try to find a second block on it. We prove that a miner with 25% computing power can get bigger revenue then.

We note as c the computing power of an attacker and G the chance to put a block in the block chain and gain the reward. If the attacker is not selfishly mining then $G = c$. If the attacker selfishly mines then

$$G = c * \left[\left(1 - \frac{1}{2^n}\right) + \frac{1}{2^n} * \left(1 - 2^{-n-1}\right) * (1 + G) \right] \quad (2)$$

where n is the additional zero bits the selfish miner's block have. The probability that the miner finds a block with less than n additional zero-bits is $\left(1-\frac{1}{2^n}\right)$. In this case the attacker decides that the block is not worth the risk and publishes it to get the reward. If on the other hand the block is good enough ($\frac{1}{2^n}$ probability) the attacker keeps it hidden and mines on it. If another miner creates a conflicting block the the attacker has $\left(1-2^{-n-1}\right)$ probability of winning the conflict, since he looses only if the other miners has found a block with $n+1$ more zeros than the target. If the miner wins he gets the block reward plus the chance to gain from the hidden blockchain (1 for the win of the conflict and G for the possibility to win the next block).

For $c = 0.25$ and $n = 2$, $G = 0.2562$ which means that if a selfish miner with 25% computing power tries to selfish mine he can get $25,62\%$ of the revenue. As n increases the minimum c decreases, thus almost everyone can try to selfish mine. Furthermore as c increases the total percentage of the revenue increases, thus mining in pools is favourable. This results leads to the same problem as the original selfish mining paper.

In ByzCoin there is only 50% chance of winning a block and in the case of loosing all hidden work is wasted. The expected wins of a selfish miner can also be calculated by equation Eq. (2) when $n = 0$. Then we can see that in order for $G > c$ we need $c > 1$ i.e the attacker needs to have more than 100% of the computing power, which is impossible.