Blockchain abbreviation: Implemented by message passing and shared memory (Extended abstract)

Conference Paper · October 2017

DOI: 10.1109/NCA.2017.8171382

2 authors, including:

Shlomi Dolev
Ben-Gurion University of the Negev
437 PUBLICATIONS **6,291** CITATIONS

SEE PROFILE

BLOCKCHAIN ABBREVIATION
Implemented by Message Passing and Shared Memory

(Extended Abstract)

Maxim Amelchenko Shlomi Dolev

Ben Gurion University of the Negev
maxam@post.bgu.ac.il, dolev@cs.bgu.ac.il

***Abstract*— Blockchain's ever increasing size has become a major problem. Bitcoin [7], for example, has grown to 115120 MB as of May 2017, which is roughly 115 GB. This uncontrollable growth of the Blockchain is bound to become an issue in the future, as hard disks may become too small to store the entire Blockchain history and traversing the transactions databases may become increasingly slow. Already, there are lightweight clients in various Blockchain platforms (Bitcoin included), who do not store the entire chain locally but rely on a third party to send them the blocks they need. There are many issues with these clients, mainly security problems, since they go back to trusting a central authority rather than gaining trust from several distributed peers. These clients' knowledge of the Blockchain is solely based on some third party that should be trusted, while the conceptual base for Blockchain is trust distributing.**

In this paper we present two Blockchain abbreviation schemes. The first one is based on the Ethereum [8] project and proposes replacing the full Blockchain with a new Genesis block, which summarizes everyone's account balances at a certain point in time. One possible benefit is to use less communication while still storing the prefix of the old Blockchain (or signature of the Blockchain that can validate a version archived by other participants) in a local archive. Here we trade loss of transaction history for efficiency. Our second contribution is a UNIX based architecture using the file system, for implementing Blockchain. We demonstrate a Blockchain abbreviation technique for this architecture too.

I. INTRODUCTION

Blockchain technology is a very promising futurist concept for constructing distributed trust structures. The main idea is to control the schedule of actions in the entire distributed system to be serial with a very high probability. The need to mine, compute and search for a hard to find string that is hashed to a specific pattern enforces even the Byzantine participants to obey the rate of new block creation that implies with high probability sequential new block creation in the entire system.

In rare cases, several blocks are introduced in parallel and symmetry is broken over time by continuous gossip among the participants and comparison of chain lengths. The gossip has a known (small) latency among all participants. Thus, allowing a short bound on the time it takes to have agreement on the finalization of a block inclusion in the block chain, allowing to finalize the (possibly, financial) transactions associated with the block.

978-1-5386-1465-5/17/$31.00

In more details, the participants in implementing a Blockchain start with a commonly agreed genesis block. This genesis block is publicly known to everyone and cannot be forged later by others since it will break the cryptographic (hash function) connection between the genesis block and the next block. They constantly perform operations which involve collecting transactions from the entire network, gathering them into a block and mining. This process of mining (which is part of a Proof-of-Work scheme on which we will elaborate later), involves repetitive hashing of the block contents (with an alternating nonce) until the hash value is smaller than an agreed upon threshold. In each round of hashing, the participants change a certain value in the block called nonce, yielding a different hash value. The first node to reach the threshold publishes its block to the entire network. Each node receiving a new block, performs a series of checks to ensure the validity of the block and the associated transactions. Once the verification is completed, the block is accepted and appended to the Blockchain in each node.

Blockchain platforms usually employ one of two methods to achieve consensus without a single point of authority. The first method is Proof-of-Work (PoW) and the other is Proof-of-Stake (PoS). We will elaborate more about PoW as it is much more common, it is easier to understand and we also utilized it in the implementation of our schemes. The idea behind PoW is making participants in a system work and be able to prove that a certain amount of work has been done, before they can use, contribute and profit from a system. PoW is used to achieve multiple goals:

- Prevent Sybil attacks. In these attacks, an attacker creates many fake identities and tries to influence the network by posing as a majority or a large subcommunity within a community. An attacker wanting to sabotage a system using PoW will have to physically own a very large amount of computational power.
- Achieve consensus using majority voting. If we represent Blockchain as a voting system, where participants vote on what is the currently active chain, then the longest chain (or the chain which required the most work to produce) is the majority vote.

- Set the data generation rate. Using PoW where the difficulty of puzzles is adjustable, allows systems to adapt to changes of its clients and maintain a stable release rate of its product (a cryptocurrency or anything else). For example, if hardware becomes stronger, or new algorithms are found to find faster solutions to puzzles, the system can increase its PoW difficulty so the data generation rate is uneffected. No special synchronization is needed to adjust difficulty, since it is calculated independently at each node based on public information like average block generation time.

The most important assumption of PoW is that there is an honest majority. More precisely, the vast majority of CPU power is controlled by honest nodes. Since the honest nodes hold the vast majority of the computational power, their chain will always be the fastest to grow and no chain will be able to compete with it.

Over the years many papers were published to address the issue of increasing Blockchain size and scalability. However, most of them targeted the general Blockchain community growth, and the scalability issues that come with it. Among them, are block size limit, databases bottlenecks and overload of network traffic. Mini-Blockchain project lead by J.D. Bruce [1] is one of the few that actually tackled the problem of the chain length. The project details a solution that involves cutting off the strong cryptographic connections between transactions and maintaining an account tree to quickly check peer balances. This allows deleting transactions that are confirmed, effectively reducing the size of the chain to a constant at the expense of keeping a large account tree. Our abbreviation scheme is conceptually different. We propose an abbreviation process that cuts a prefix of one or more blocks from the Blockchain, without altering the connections between the remaining blocks. A more formal definition of the abbreviation will be given for each one of the schemes we present.

II. Blockchain System Settings

We define Blockchain as a distributed system of machines, each running the same program consisting of executable statements. We represent this distributed system by a set of n state machines called nodes that communicate with each other. Every node can communicate with its neighbors either by exchanging messages, or using shared memory. When using shared memory, all nodes must know one another, forming a complete undirected graph of nodes. When using message passing, each node has to be connected to at least one other node, and a (reliable, non-Byzantine) path must exist between every two nodes, forming an undirected tree graph.

Communication by writing in, and reading from, the shared memory usually fits systems where the nodes are geographically close to each other. A message-passing distributed model fits both nodes that are located close to each other and wide-area distributed systems such as communication networks. In a message-passing model nodes communicate by sending and receiving messages over the wire. In such a system, network latency can have a big influence over the synchronization model of the system. This has been examined to some extent by Garay, Kiayias and Leonardos [4]. We will further elaborate how it affects the Blockchain, taking into account the abbreviation times.

III. Preliminary

It has been proved that for any distributed system of size n, for consensus to be reached, no more than $\frac{n}{3}$ nodes can be faulty [3] . Faulty processes can exhibit a number of behaviors, such as presenting different messages to different nodes, dropping messages, and altering messages on transit. However, Blockchain, which is also a distributed system, does not have these limitations. The reason lies in the mining process, and in the ability to verify the correctness of newly published blocks. In order to distribute two different but legitimate blocks, a node will have to beat the entire network twice in a row thus possessing an unrealistic amount of computational power. Even if a node manages to ditribute two different but legitimate blocks, causing a fork in the network, eventually when more blocks are added the network will converge to a single consensus chain, based on the difficulty of all the blocks. Once a block is published, all other nodes can verify the transactions in it are valid, and that enough work has been put into creating this block. This guarantees that messages accepted in the Blockchain are legitimate.

IV. Message Passing Blockchain Abbreviation

Overview. Message Passing Blockchain is a type of Blockchain where nodes send each other messages. This Blockchain is usually cryptocurrency-based, but other use-cases have also been demonstrated, such as voting systems [6]. In this architecture, not every node has to know every other node but there must exist a channel between each node to at least one other node. Nodes joining the network have a number of ways of discovering their peers. Over the years, IRC (Internet Relay Chat) channels have been used to locate IP addresses of running nodes. When joining IRC channels, the client receives messages broadcasted from other clients already subscribed to that channel. This method has been abandoned in favor of DNS seeds and hard-coded IP lists. The DNS seeds resolve to a group of IP addresses which are known to be stable. If a node is experiencing a DNS failure it will try the hard-coded IP addresses.

We will address the common situation where an asset-based Blockchain (such as cryptocurrency) uses the message-passing model for communication. In this scheme, every participant has an account which contains its balance of assets (for example, number of coins). We will use this feature in the abbreviation of the chain, as capturing the balances of all participants at a certain point in time can serve as a reference point for all future transactions. This point in time can become the new “beginning of time", or the new Genesis block.

(a) Message-passing model

(b) Shared-Memory model

Figure 1: Communication models in Blockchain

Abbreviation. Let B be a Blockchain consisting of blocks $B_1, ..., B_n$, where B_1 is the genesis block. Abbreviating this chain is replacing it with a new genesis block B_g which summarizes the account balances of all participants.

Network Latency. In our context, we define network latency as the maximal amount of time between a block publication and the last node in the network receiving this block. The network latency has a direct effect on the Blockchain abbreviation, since it might influence whether an abbreviated Blockchain gets accepted by the vast majority or not. If a new block is mined faster than the time is takes to abbreviate a chain it might lead to a situation where abbreviated chains never get accepted. This might happen because by the time the new abbreviated chain reaches everybody it will not be up to date as other nodes would have received the next block which is not included in the abbreviation. We present two lemmas in connection with network latency. The first lemma deals with the *uniform latency case*, in which the rate of information propagation is at least a linear function of the elapsing time. Namely, there is a function of time (since a message is sent) to the percentage of system nodes that receive the message. This is similar to what happens in a network of nodes spread out across a single LAN, in a small geographical location such as a university campus. The second lemma deals with non-uniform network latency which defines an unpredictable rate of propagation through the network. Namely, there is no information on the portion of the nodes that received the message following the transmission of the message, but there exists a bound on the time it takes for all nodes to receive the message. This is more relevant in a network spread out across a large geographical area (WAN), such as the Ethereum network.

Uniform Network Latency. Uniform Latency is defined as uniform propagation through the network. This means that a message propagates through the network in a predictable manner. Formally let l be the network latency in a uniformly latent network of size s, and let x be a constant in the range $0 \dots 1$. Suppose a block is published in this network. After $x \cdot l$ time, at least $x \cdot s$ nodes in the network have received the block.

Lemma IV.1. *Assuming uniform latency through the network, abbreviation of the Blockchain must take less time than the time it takes to mine a new block. Otherwise, the abbreviated chain may never be accepted and the network may continue to mine on the original chain.*

Proof Sketch. Assume that the lemma is not true. For the sake of simplicity, assume that mining takes a constant time of x and abbreviating a chain takes a constant time of $y > x$. Suppose a block b_1 is published at time t_1 and assume a best case scenario where node n_1 starts abbreviating the chain (including b_1) at time t_1. Now suppose a block b_2 is published at time $t_2 = t_1 + x$. n_1 is not done abbreviating yet, it will finish at time $t_3 = t_1 + y > t_2$. b_2 will start propagating through the network before the new abbreviated Blockchain is released. Uniform latency ensures a vast majority of the nodes will receive the block b_2 before receiving the new chain. When the new chain will eventually reach everyone, it will be discarded by the vast majority because it will not be up to date (it will not include transactions from b_2). Nodes will continue mining on the original chain including the newly published block b_2. So y cannot be larger than x for abbreviation to work.

□

Non-Uniform Network Latency. Non-Uniform Latency is defined as unpredictable propagation through the network. This means that there is no guarantee on the percentage of nodes that receive a message after a certain amount of time. Formally, let l be the network latency in a non-uniformly latent network of size s. Suppose a message is transmitted in this network. After m time passes, where $m < l$, there is no guarantee on the amount of nodes that have received the message. Note, that when l time has passed, the entire network has received the message.

Lemma IV.2. *Let l be the network latency in a non-uniformly latent network. Let t be the time it takes to mine a block and $t > l$. Then Abbreviation time must take less than $t - l$.*

Otherwise, the abbreviated chain may never be accepted and the network may continue to mine on the original chain.

Proof Sketch. Assume that the lemma is not true. For the sake of simplicity, assume that abbreviating a chain takes a constant time of y where $t > y > t-l$. Suppose a block b_1 is published at time t_1 and assume a best case scenario where node n_1 starts abbreviating the chain (including b_1) at time t_1. Now suppose a block b_2 is published at time $t_2 = t_1 + t$. n_1 is already done abbreviating at this point but the abbreviated chain has not reached all nodes yet because the network latency l ensures it will reach all nodes at time t_1+y+l. Notice that the following holds: $t_1 + y + l > t_1 + t$. Since there is no guarantee on the propagation rate of b_1, it is possible that more than half of the nodes receive it by the time the abbreviated chain reaches half of the network. In that case, the vast majority of the network is going to mine on the new block b_2 and not on the abbreviated chain. So abbreviation time must take less than $t - l$. □

A. Blockchain protocol changes.

Currently the rule of Proof-of-Work in different Blockchain platforms is that if a node encounters multiple chains it must mine on the one which required the most computational power to produce (generally the longest one). If there are several chains with equal length the node may choose arbitrarily on which chain to mine and eventually the network will settle on a single longest chain which everyone will adopt. This rule conflicts with Blockchain abbreviation since abbreviating means shortening the chain. So how will nodes know the shorter chain needs to be adopted? The solution to this is modifying the difficulty of the genesis block to be the sum of difficulties of all blocks summarized by the abbreviation. If we are abbreviating a chain of three blocks $A \to B \to C$, each of difficulty 10, we create a new chain consisting of one genesis block G of difficulty 30. This difficulty is only symbolic, since it does not represent the amount of work spent on mining this new genesis block, but the amount of work spent on all blocks summarized by this genesis block. The protocol must be changed so nodes do not verify the hash of the genesis against its difficulty. The genesis hash will not match the claimed difficulty. Another minor change is that participants must allow the genesis block to increase in size beyond the configured limits (usually Blockchain platforms limit the size of published blocks).

The Blockchain protocol needs to change so everyone in the network must accept an abbreviated Blockchain once such a chain is available. Otherwise, the system is susceptible to attack from Byzantine nodes, which can generate much more revenue than they are supposed to.

Claim IV.3. *Formally, consider three groups of nodes mining together on a Blockchain. Group F (Fast), adopt an abbreviated chain once it is released. Group S (Slow), want to keep the transaction history, so they continue to mine on the unabbreviated chain for a few blocks, in hopes of taking over the abbreviated chain. Eventually, they comply and switch. Group B (Byzantine), abbreviate the chain after every new block, and try to mine on it. The computational power of the three groups F, S and B are f, s and b respectively. If group B is large enough that together with another group they constitute a vast majority, group B can generate more revenue than they are supposed to.*

Proof Sketch. Formally, consider three groups of nodes. Group F (Fast), adopt an abbreviated chain once it is released. Group S (Slow), want to keep the transaction history, so they continue to mine on the unabbreviated chain for a few blocks, in hopes of taking over the abbreviated chain. Eventually, they comply and switch. Group B (Byzantine), abbreviates the chain after every new block, and tries to mine on it. The computational power of the three groups F, S and B are f, s and b respectively. Suppose these inequalities hold: $s > b > f$ and $s < f + b$. Here we assume group B is the middle group in size, but the lemma is true even if it is the smallest group in the network. Since the group s does not adopt abbreviated chains right away, they never mine on the chain the vast majority is mining on. The groups B and F constitute a vast majority and they get all the revenue. The fair share of the Byzantine group is $\frac{b}{b+s+f}$. However since they split the revenue with the Fast group their actual revenue is $\frac{b}{b+f}$. Because $b > f$, the Byzantines have actually majority control though they do not form a vast majority in the entire network. Besides generating more revenue than they should, the B group can even drop blocks mined by the F group and publish only their own blocks. Since they are bigger in size than the F group they are guaranteed to take over any chain of any length published by group F. For this reason, an abbreviated Blockchain needs to be accepted by all honest nodes. We suggest including a (strong) cryptographic hash (e.g. SHA-3) result of the entire abbreviated blocks in the new genesis block, as a proof for the original block values. Thus, ensuring proof of continuation. □

Ethereum based implementation. As a proof of concept, we implemented the above schema using the open source Ethereum project, which is the code driving the Ether digital coin. Obviously, we could not mine on the public branch since we wanted to test the abbreviation and also guarantee short mining time. We used Ethereum's private chain functionality to create our own chain. We set the difficulty to be very low to guarantee successful mining within a short period of time (a few seconds). Next, we created two virtual machines that would act as mining nodes (e.g., peers, as described by the Ethereum protocol) and started mining on both. Once enough blocks have been mined, we stopped the mining process on both virtual machines, and planted a new Genesis block on each machine. The new Genesis block was similar to the default one given by the Ethereum project, but it also had the balances of the two nodes as captured the moment they stopped mining. This was possible due to the new 'alloc'

directive defined in Ethereum, which cannot be found in all cryptocurrency projects yet. Next, we manually deleted the chain directory from both machines (effectively deleting all blocks from the Blockchain) and started mining again. We verified that both machines continued to generate revenue on top of their preallocated balances and that the block count started from the beginning again.

V. Shared Memory Blockchain Abbreviation

Overview. Shared Memory Blockchain is a type of Blockchain where nodes connect to one another to maintain Blockchain consistency, rather than send each other blocks they have mined. This Blockchain is not cryptocurrency-based, but deals with publishing records (or arbitrary data) in a public ledger, therefore no accounts or wallets are required. This is crucial since when we introduce Blockchain abbreviation, the lack of miner state will greatly affect the procedure.

Assumptions:

- ***Fixed pool of miners***: A pool of fixed size where everyone knows everyone.
- ***Honest Majority***: Byzantine nodes can comprise up to half of the entire network.
- ***No miner state***: No record is kept of each miner, such as balance, wallets, history of transactions, etc.
- ***Abbreviation is decided by majority vote***: Byzantine nodes are included, meaning that if a group of honest nodes and a group of Byzantine nodes together constitute a vast majority, their vote to abbreviate the chain or not will be decisive.
- ***No down time***: Nodes do not disconnect from the network.

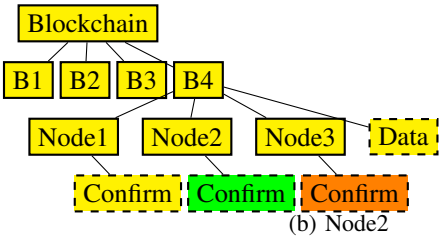
Genesis Block. The Genesis block in this Blockchain is simply the earliest block ever mined. If a certain prefix of the Blockchain is deleted then the Genesis block is simply the earliest one that exists.

Mining. Nodes continue to mine as before, where they need to find a hash smaller than a certain threshold in order to be able to publish their data.

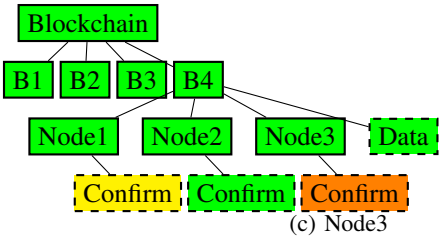
Blockchain Structure. The Blockchain is a directory structure stored on each node separately. Each miner has a user account on every other nodes' machine. All miners must belong to the same UNIX group "miners". The structure is as follows:

- A general Blockchain directory - owner root, group miners,
  UNIX permissions: 770
- A directory for each block - owner root, group miners,
  UNIX permissions 770
- A directory for each miner - owner minerX, group miners,
  UNIX permissions 740
- A file confirming validation of this block - owner minerX, group miners,
  UNIX permissions 700

(a) Node1



(b) Node2



(c) Node3

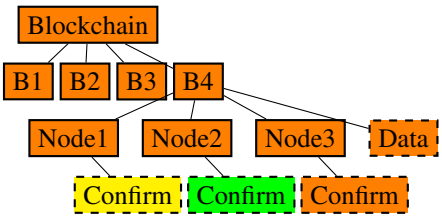


Figure 2: A Shared Memory Blockchain with three nodes. A total of four blocks have been mined. Dashed boxes are files. All other boxes are directories.

The structure makes sure the Blockchain at each machine is open for all users belonging to group "miners", but no user can interfere with another user's validation due to separation of directories between users, and Read privileges only given to the group.

Block. A block is a single unit of data in the Blockchain. Usually it is complex and contains lots of fields but for our purpose this structure is enough:

- Serial Number
- Miner id
- Previous Block hash
- Nonce
- Data

The Genesis Block is the same, except that it may contain arbitrary data in the Previous Block hash, since there is no previous block.

Abbreviation. Let B be a Blockchain consisting of blocks $B_1, ..., B_n$. Abbreviating this chain at abbreviation point t is cutting the prefix of $B_1, ..., B_t$ from this chain to produce the new abbreviated chain: $B_{t+1}, \ldots, B_n$.

Confirmation. Nodes regularly connect to all other nodes, and write confirmation files into each block in their Blockchains. They are used when nodes want to vote for Blockchain abbreviation and delete their files from a particular block. Nodes

confirm blocks sequentially, so block x will be confirmed before block $x+1$.

Nodes constantly look if their chains need to be abbreviated. Consider a network of size n. Each node searches its chain for a maximum block x, such that block x has less than $\frac{n}{2}$ confirmations, while block $x+1$ has more than $\frac{n}{2}$ confirmations. This ensures that the missing confirmations in block x were indeed erased and not just delayed by network latency. Once such a block is found, all blocks preceding and including x can be erased. No further changes are required since the Previous Block Hash of the now new genesis block $x+1$ is unchanged and so the hash of the block x remains also unchanged. So block $x+2$ Previous Block Hash is still correct and this propagates to the rest of the blocks.

A. Blockchain Consistency Among Byzantine Nodes

Byzantine nodes have been a part of Blockchain research from the very beginning. Having to decide on a single, main chain is similar to the Byzantine Generals problem [5], where a group of army generals must decide on a strategy in the presence of traitors. The assumption of honest majority has played an important part of mitigating attacks from Byzantines. In particular, nothing can effect the Blockchain if only a minority of the network deviate from the protocol. As long as the vast majority continue to mine and publish blocks as always, they will control the network. The most prominent attack from malicious nodes is selfish mining, which was detailed in a paper by Eyal and Sirer [2]. But this attack did not have an effect on the consistency of the Blockchain, only on the revenue of the attackers. By introducing abbreviation, we also introduce weak points which can undermine the consistency of the Blockchain.

If a minority of honest miners want to abbreviate the chain at a certain point, they delete their confirmations from the corresponding block in all nodes. Suppose there exists a group of Byzantine nodes which delete their confirmations from the same block as the honest nodes but only from a part of the entire network. If the honest nodes which voted for abbreviation and the Byzantine nodes together constitute a vast majority this may lead to a situation where a part of the network mine on an abbreviated chain while another does not. There needs to be a mechanism for converging to one single chain when such inconsistencies occur. In **Algorithm A1**, we show how nodes keep track of whether they are out of synchronization with the majority of the network.

Let n denote an arbitrary honest node in the network. While connecting to other nodes to write confirmations, n counts how many nodes in the network mine on a chain which starts with a different Genesis block. n also saves all the last blocks from all the chains in the network. If more than half of the nodes have a different Genesis block than n, and the median of all last blocks in the network is ahead of the last block of n by a predetermined value, then the node assumes it has fallen behind on its chain, and needs to catch up with the majority. This predetermined value depends on the specific Blockchain platform. For example in Bitcoin, it is generally accepted to wait for six blocks before considering a transaction to be confirmed. Probabilistic analysis has shown that the chances for a chain to be overtaken after it has advanced six blocks over a given block are negligible. This value of six can be adopted for our purposes too.

We look at the median of all last blocks since that way all the Byzantine nodes cannot declare they have an extremely high or low serial number in the last block and manipulate the network into making decisions based on that. When the median block is significantly ahead of our block it is safe to say that the vast majority is mining on a different chain, since our chain is not catching up with it. Next, we present a lemma which shows that Byzantine nodes cannot undermine the consistency of the Blockchain.

Lemma V.1. *Let n be a network of miners, divided into two groups with mining power g_1 and g_2. Without loss of generality, suppose $g_1 = g_2 \cdot y$, and $y > 1$. Let us assume they mine on two separate chains starting at length 0. After sufficient time passes, g_1 will have x more blocks than g_2, for any $x > 0$.*

Proof Sketch. Since g_1 has more computing power, over time they will commit more blocks than g_2 by a factor of y. After a certain period of time g_1 will have $b_1 \cdot y$ blocks on its chain while g_2 will have b_1. Thus $b_1 \cdot y - b_1 = x$ and this yields $b_1 = \frac{x}{y-1}$. So after g_2 mines $\frac{x}{y-1}$ blocks it will be x blocks behind g_1.

□

Lemma V.2. *A Byzantine group of nodes z cannot undermine the consistency of the Blockchain by erasing confirmations from a subgroup of nodes t of the whole network n, $t < n$.*

Proof Sketch. Since the group z is less than half of the entire network (by the honest majority assumption), they can only succeed in making nodes abbreviate their chain if more legitimate nodes also vote for abbreviation. Let l be a group of legitimate nodes that want to abbreviate the chain at some block b_1. Assume that $l < \frac{n}{2}$ and $z < \frac{n}{2}$ but $l + z > \frac{n}{2}$. Since l are honest they erase their confirmations at block b_1 in the entire network but z erase their confirmations only from a group of nodes t.

Case 1: $t > \frac{n}{2}$ – the minority is mining on the original long chain.

In this case more than half of the network abbreviates their chain so the rest of the network falls behind and continue to mine on the original chain. Let m denote the minority group. By **Algorithm A1**, the group m will continue to mine until the median of the serials of the last blocks in the network is ahead of their chain by a predetermined x blocks. This is bound to happen by the previous lemma. After that happens, m will switch chains to a longer chain. This chain is going

to be verified by Proof-of-Work **Algorithm A2** so Byzantine nodes cannot cheat nodes into adopting inconsistent chains.

Case 2: $t < \frac{n}{2}$ – the minority is mining on the new abbreviated chain.

In this case less than half of the network abbreviates their chain so this group fall behind and mine on a separate chain. Let m denote the minority group. By **Algorithm A1**, the group m will continue to mine until the median of the serials of the last blocks in the network is ahead of their chain by a predetermined x blocks. This is bound to happen by the previous lemma. After that happens, m will switch chains to a longer chain. This chain is going to be verified by Proof-of-Work (**Algorithm A2**) so Byzantine nodes cannot cheat nodes into adopting inconsistent chains.

□

Implementation over Unix file system. As a proof of concept, we implemented a complete Shared Memory Blockchain framework using the Java language. We used three small virtual machines, each running a mining node written in Java and an SSH server for communication between the nodes. Each virtual machine contained a user group called "miners", and three user accounts, one for each node. We created a directory structure in each machine, similar to that described above, where every block in the chain corresponds to a directory. In each such directory there are three more directories, one for each node, and a file containing the data in that block. The data published by nodes was an arbitrary 256 bit string generated by a random function. For mining we implemented a scheme similar to other Blockchain systems, where nodes have to compute a hash smaller than a certain value in order for it to be valid. Periodically, each node would connect to every other node, and write confirmation files to their designated directory. During this communication, if a machine notices that another machine has a more advanced Blockchain it will copy it, after verification, and resume normally. For maintaining consistency, we used **Algorithm A1**, and ran some manual tests by erasing confirmation files. All tests showed that all nodes eventually converged to a single Blockchain, and continued to mine normally.

Algorithm 1 Maintaining Blockchain consistency by median of all last blocks in the network - here we set a constant of 6 for deciding that we are out of sync with the network:

1: **Loop:**
2: SYNC=false //Is this node out of sync
COUNT=0 //Amount of nodes out of sync with this node
LATENCY=6 //network latency
BLOCKS=[] //All latest blocks from all nodes

3: **for each node in network:**
4: **if** $node.blockchain.firstblock \neq blockchain.firstblock$ **AND** $\neg SYNC$ **then**
$COUNT = COUNT + 1$
$BLOCKS.add(node.blockchain.lastblock)$
5: **else** $getChain()$

6: $medianblock = BLOCKS.sort().get(network.size()/2)$

7: **if** $COUNT \geq network.size()/2$ **AND** $medianblock - blockchain.lastblock \geq LATENCY$ **then**
$SYNC = true$

Algorithm 2 Verify Blockchain correctness:

1: **procedure** VERIFYBLOCKCHAIN(BLOCKCHAIN)
difficulty = 6
for each B in BLOCKCHAIN:
2: **if** $B.hash \neq SHA256(B)$ **then** $return false$
3: **if** $B.prevhash \neq prev(B).hash$ **OR** $prev(B).serial + 1 \neq B.serial$ **then** $return false$
4: **if** $B.hash.substring(0, difficulty) \neq 000000$ **then** $return false$
$return\ true$

ACKNOWLEDGEMENT

We thank Yossi Gilad and Sergio Rajsbaum for comments and corrections.

REFERENCES

[1] J.D. Bruce The Mini-Blockchain Scheme (2013). *Available at* https://cryptonite.info/files/mbc-scheme-rev3.pdf.
[2] Ittay Eyal, Emin G. Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable (2013). *Available at* http://arxiv.org/abs/1311.0243.
[3] Michael J. Fischer, Nancy A. Lynch and Michael Merritt. Easy impossibility proofs for distributed consensus problems (1986). *Available at* https://link.springer.com/article/10.1007/BF01843568.
[4] Juan A. Garay, Aggelos Kiayias and Nikos Leonardos. The Bitcoin Backbone Protocol:Analysis and Applications (2017). *Available at* https://eprint.iacr.org/2014/765.pdf.
[5] Leslie Lamport, Robert Shostak, and Marshall Pease The Byzantine Generals Problem (1982). *Available at* www.microsoft.com/en-us/research/wp-content/uploads/2016/12/The-Byzantine-Generals-Problem.pdf.
[6] Patrick McCorry, Siamak F. Shahandashti and Feng Hao. A Smart Contract for Boardroom Voting with Maximum Voter Privacy (2017). *Available at* https://eprint.iacr.org/2017/110.pdf.
[7] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System (2008). *Available at* https://bitcoin.org/en/bitcoin-paper.
[8] Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger (2014). *Available at* https://ethereum.github.io/yellowpaper/paper.pdf.