Brief Announcement: Bootstrapping Public Blockchains Without A Trusted Setup

Abhinav Aggarwal  
University of New Mexico  
Albuquerque, NM 87131  
abhiag@unm.edu

Mahnush Movahedi  
Dfinity  
Palo Alto, CA  
mahenush@gmail.com

Jared Saia  
University of New Mexico  
Albuquerque, NM 87131  
saia@cs.unm.edu

Mahdi Zamani  
Visa Research  
Palo Alto, CA  
mzamani@visa.com

ABSTRACT

We propose a protocol that allows the participants of a permissionless decentralized system to agree on a set of identities in the presence of a computationally-bounded Byzantine adversary. Our protocol guarantees that the fraction of identities belonging to the adversary in the set of identities is at most equal to the total computational hash power of the adversary.

We significantly improve on the existing state-of-the-art in the following four ways. First, our algorithm runs in expected $O(1)$ rounds, in contrast to previous results which require $O\left(\frac{\log n}{\log \log n}\right)$ rounds, where $n$ is the number of initial nodes in the system. Second, we require each node to solve only one computational puzzle, whereas previous algorithms require $O\left(\frac{\log n}{\log \log n}\right)$ puzzles per node. Third, our algorithm sends only $O(n)$ bits per node in expectation, whereas previous algorithms send $O\left(\frac{n \log^2 n}{\log \log n}\right)$ bits in expectation. Finally, in contrast to past results, our algorithm handles dynamic joining and leaving of nodes.

1 INTRODUCTION

Blockchain protocols rely on an agreement mechanism to ensure that their participants collectively decide on the next block of transactions. Current protocols require some trusted initial setup to defend against Sybil attacks [6], where an adversary maliciously influences the collective decisions of the system by generating a large number of fake identities. One technique is to use a genesis block [16], which ensures that all nodes begin proof-of-work [11], or PoW, puzzles at the same time [13]. Another technique is to assume a public-key infrastructure (PKI) [15, 18], which provides authenticated communication.

In this paper, we consider an adversary controlling $f < \frac{1}{2}$ fraction of the computational power in the network. We ask: Can we efficiently enable agreement on a set of identities against this adversary, without any trusted setup? In particular, can we ensure that the participants agree on a set of identities, where the fraction of adversarially controlled identities in this set is at most $f$?

Aspnes et al. [4] first formally addressed this problem using PoW puzzles, but with no genesis block. Subsequent work [3, 13, 14, 17] improved efficiency and robustness of this initial solution. The current state-of-the-art, by Hou et al. [14] solves this problem in $O\left(\frac{\log n}{\log \log n}\right)$ rounds, where in each round, every honest node needs to send $O(n \log n)$ bits and solves one computational puzzle. We improve this result as follows.

**Theorem 1.1 (Informal).** Our algorithm ensures that $n$ participants agree on a set of identities such that the fraction of identities in this set controlled by an adversary is at most equal to its computational hash power. Moreover, (1) each node solves a computational puzzle only once; (2) the protocol runs in $O(1)$ rounds with high probability; (3) a total of only $O(n)$ bits per node, in expectation, are sent; and (4) a linear amount of dynamic joins and leaves from the system is allowed with only $O(n)$ additional messages per node.

Our Model. We assume a synchronous network of $n$ nodes $P_1, \ldots, P_n$, where $n$ is initially unknown. All messages are exchanged using a diffuse primitive, which enables a message to be sent to all other nodes. Without loss of generality, we assume that all honest nodes have the same computational power. Additionally, we assume a Byzantine adversary who controls up to an $f < \frac{1}{2}$ fraction of the total computational power. We assume that the adversarially controlled nodes may deviate from the protocol in any arbitrary
Node $P_i$ (with set $S_i$ and $n_i \leftarrow 2^{\lceil \log |S_i| \rceil}$) does the following:

1. **Committee Election**. Arrange all solutions $s_j = H(h_j | pk_j | C_j)$ into buckets of size $\frac{1}{n_i}$, so that $s_j$ falls into the $k^{th}$ bucket $b_{i,k}$ iff $s_j \in \left( \frac{k-1}{n_i}, \frac{k}{n_i} \right]$. Let $B_{i,k}$ denote the set of solutions that fall into bucket $b_{i,k}$. Let $f_{i,k} = \arg \min_j (s_j \in B_{i,k})$ be the public key of the node with the smallest solution in the $k^{th}$ bucket and $CView_i = \bigcup_{k=1}^{c} \{ f_{i,k} \}$.

2. **Byzantine agreement**. If $pk_j \in CView_i$, run the Byzantine agreement algorithm of Abraham et al. [1] using input $S_i$ with other members in $CView_i$. Diffuse the output set to the entire network.

3. **Final Output**. If $pk_j \notin CView_i$, output the set obtained using a majority filtering from the sets of the nodes in $CView_i$.

![Figure 1: Our View-Reconciliation Protocol](image)

manner. We do not rely on any trusted setup or secure broadcast channel, but we assume the existence of a random oracle hash function [11, 14]. We finally assume up to an $\epsilon < 1/6$ fraction of the nodes can join or leave after the bootstrapping is complete.

**Rounds.** Our algorithm proceeds in rounds, in which a node can perform the following three steps: (1) receive messages from other nodes; (2) perform some local computation; and (3) send messages to other nodes. Following past work [14], we assume all local computation except solving puzzles is instantaneous.

### 1.1 Solution Overview

Our protocol consists of two phases. In the first phase, all nodes perform an initial diffusion of computational puzzles, similar to the protocol of Aspnes et al. [4]. In the second phase, we propose a novel solution to the view-reconciliation problem [14] in order to resolve inconsistencies among the views of the honest nodes.

**Phase I (Sybil Defense).** The first phase proceeds as follows: (1) Each honest node $P_i$ locally generates a random public/private key pair $(pk_i, sk_i)$ along with a random challenge string $c_i \in \{0, 1\}^k$, where $k$ is the security parameter. This challenge is then diffused to the network. (2) Let $C_i$ denote the set of challenges that $P_i$ receives from other nodes, and $H$ be a random oracle hash function known to all the nodes. $P_i$ attempts to solve a PoW puzzle by computing a nonce $h_i \in \{0, 1\}^k$ such that $H(h_i | pk_i | C_i) < d$, where $d$ is the PoW difficulty parameter (determined similar to [14]). Once a valid solution is obtained, $P_i$ diffuses the tuple $(h_i, pk_i, C_i)$ to the network. (3) For every tuple $(h_j, pk_j, C_j)$ received, $P_i$ checks if the PoW was computed correctly with respect to $h_j, pk_j, d$, and that $c_j \in C_j$. If so, $P_i$ includes $pk_j$ in its local identity set $S_i$ and sets $n_i \leftarrow 2^{\lceil \log |S_i| \rceil}$ as its estimate of the number of unique identities.

The above algorithm ensures that the local identity sets of honest nodes will contain all honest identities and at most $f$ fraction of the identities from the adversary. However, it is possible for these sets to be inconsistent in the identities they contain. Hence, in the second phase, we need to ensure that the honest nodes agree on a single, common set of participants in the system.

**Phase II (View Reconciliation).** We now describe our view reconciliation algorithm through which honest nodes can agree on exactly the same set of identities. The main steps of this phase are described in Figure 1.

**Step 1) Committee Election.** Note that the PoW solutions obtained so far have sufficient randomness [5] to locally select a random committee without any communication among the nodes. Thus, we can perform a probabilistic committee election to achieve Byzantine agreement on the set of identities. We run agreement among only $O(\log n)$ identities to limit the number of bits exchanged during this phase.

Our committee election protocol proceeds as follows. Each node $P_i$ splits its local set $S_i$ into $n_i$ buckets, where the identity with the smallest solution in each bucket is then chosen as the representative of that bucket. The set of the first $c\log n_i$ bucket representatives constitute $P_i$’s local view $CView_i$ of what it considers as the committee. Since the adversary can send messages to a strict subset of the nodes, it is possible that $n_i \neq n_j$ and $CView_i \neq CView_j$, for some honest node $P_j$. Our algorithm ensures that with probability at least $1 - O\left(\frac{\log n}{n}\right)$, the views of the committee members can differ only in the membership of the adversarial nodes and contain a sufficiently-large core of honest identities. This is sufficient to run Byzantine agreement among these nodes [9, 10].

Moreover, the solutions computed by the adversary must fall uniformly at random into the buckets of the honest nodes. This is because for an honest node to accept a solution $s$, the corresponding puzzle must have included the honest node’s challenge string from Phase I. Thus, the adversary could not have precomputed a solution to the puzzle and hence, $s$ is uniformly random in its range (by the random oracle assumption).

**Step 2) Byzantine agreement.** The agreement on the set of honest identities in the local $CView_i$ at the end of Step 1 allows for the use of Byzantine agreement protocol by Abraham et al. [1] as a subroutine executed by the committee members to decide on the final set of identities. This protocol is able to handle the selective message sending by the adversary and allows the committee members to agree on the membership in the system.

Since the committee size is $O(\log n)$, in expectation, only $O(\log^3 n)$ bits per committee member are sent for the view-reconciliation. Additionally, since each honest node is equally likely to be in the committee, this bandwidth cost is load balanced in expectation.

**Step 3) Diffusing the Final Output.** Once the committee members have reached an agreement on a final set of identities, they diffuse this solution to the network. Each node then takes the set received from a majority of nodes in the committee as their final output.

### 1.2 Handling Non-Simultaneous Joins and Linear Churn

In a permissionless setting, nodes may join or leave the system at various times. Moreover, the initial $n$ nodes that run the protocol may join the system at different times. To handle this type of
We suspect that handling this case is non-trivial in that simply replacing the Byzantine agreement protocol with a version that is efficient in terms of its simplicity, bandwidth and number of solutions to PoW puzzles required by the nodes. In expectation, we terminate in a constant number of rounds, require each honest node to solve only one computational puzzle and send only \(O(n)\) messages per node, until the new committee is elected and with high probability, the system always maintains an honest majority and a consistent set of identities at any time.

2 CONCLUSION AND FUTURE WORK

We have described a protocol that allows a set of nodes to agree on a set of identities for each other such that the number of Sybil identities is minimized. Compared to past work, our algorithm is efficient in terms of its simplicity, bandwidth and number of solutions to PoW puzzles required by the nodes. In expectation, we terminate in a constant number of rounds, require each honest node to solve only one computational puzzle and send only \(O(n)\) bits per node.

Some interesting problems for future work are as follows. (1) Is there a lower bound on the number of bits required for view reconciliation for permissionless systems? (2) What happens if the adversary performs adaptive or sporadic corruption of nodes? We suspect that handling this case is non-trivial in that simply replacing the Byzantine agreement protocol with a version that is robust to sporadic participation will not suffice. Finally, (3) although it has been shown that PoW based schemes cannot be used for asynchronous networks [2], is it possible to establish a trusted setup in such systems?

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation grants CNS 1816250, CCF 1320994 and CNS 1318880. We also thank Dr. Loi Luu from Kyber Network for his insightful comments and discussions during the early phases of this work.

REFERENCES