Proof of Work Without All the Work

Diksha Gupta
Dept. of Computer Science, University of New Mexico, NM, USA
dgupta@unm.edu

Jared Saia[1]
Dept. of Computer Science, University of New Mexico, NM, USA
saia@cs.unm.edu

Maxwell Young[2]
Dept. of Computer Science and Engineering, Mississippi State University, MS, USA
myoung@cse.msstate.edu

Abstract

Proof-of-Work is an algorithmic tool aimed at securing networks by enforcing participating devices to perform computational work. A major deterrent to widespread use of proof-of-work is its continual need for solving computational puzzles, even when the network is not under attack.

In the present work, we address this issue by designing an algorithm with the following properties. First, we guarantee that a majority of devices in the network is correct at any time i.e., the network is always secure. Second, we optimize the computational cost to the correct devices. This cost is a linear function of the number of devices that ever join the network plus the computational cost of an attacker, if any.

Our results hold in a dynamic, decentralized system where participants join and depart over time, and where the total computational power of the attacker is up to a constant fraction of the total computational power of correct devices.

2012 ACM Subject Classification Computing methodologies → Distributed algorithms; Security and privacy → Security protocols

Keywords and phrases Proof of Work, Sybil Attack, Byzantine Consensus, Committee Election, Peer-to-peer networks, Bitcoin

Digital Object Identifier 10.4230/LIPIcs.SCNDS.2018.

Category Short Paper

Related Version Full version hosted on `https://arxiv.org/abs/1708.01285`.

1 Introduction

Twenty-five years after its introduction by Dwork and Naor [10], ***proof-of-work (PoW)*** is enjoying a research renaissance. Originally, PoW was conceived of as a technique for preventing malicious users from acquiring more than their "fair share" of a system resource such as bandwidth or a server's computational power. In recent years, PoW provides a critical primitive for cryptocurrencies such as Bitcoin [27], along with other blockchain technologies such as Ethereum [13], BlockStack [2], and Chain Incorporated [21, 30].

[1] The work of the first and second authors is supported by the National Science Foundation grants CNS-1318880 and CCF-1320994.

[2] This work is supported by the National Science Foundation grant CCF 1613772, CNS 1816076, and by a research gift from C Spire.

Yet, despite success with Bitcoin and its analogs, PoW has not fulfilled its promise of mitigating a wider range of malicious behaviors such as application-layer distributed denial-of-service (DDoS) and Sybil attacks [8]. These attacks are well-known and enduring security problems for which PoW seems well-suited, and yet proposals [31, 34, 22, 16, 6, 35, 26, 4, 24] built around PoW have seen only limited deployment.

A Barrier to Widespread Use. A major impediment to the widespread use of PoW is "the work". Current PoW schemes require a significant expenditure of resources to secure a system, *even when the system is not under attack*.

Cryptocurrency systems have cleverly provided monetary incentives for performing the computational work necessary to ensure PoW-based security. They have been successful, both in terms of practical impact [12, 9, 7, 25, 20], and research impact [14, 33, 3, 27, 15].

However, the perpetual resource burning inherent to systems like Bitcoin is undesirable for several reasons. First, *energy consumption:* in 2015, the Economist calculated that Bitcoin consumes at least 1.46 terawatt-hours of electricity per year, or enough to power $135,000$ homes [11]. This has significant environmental and economic impact that will increase as technologies like Bitcoin become more widely used. Second, *scalability:* high energy consumption prevents using current PoW approaches on the many other large open systems that also require security. Third, *feasibility:* in networks of battery-powered devices – for example, in many ad-hoc wireless settings – energy must be used sparingly, so current PoW systems are simply infeasible. Finally, *security:* when the mechanism that provides security is expensive, agents will likely selfishly seek to reduce costs and thereby compromise security.

Given these problems, we seek to reduce these costs by focusing on the following question: **Can we design PoW systems where the resource costs are low in the absence of attack, and grow commensurately with the effort expended by an attacker?**

In this paper, we describe an algorithm that answers this question in the affirmative (refer [19] for the full version). We present our result in the context of the Sybil attack; however, it applies more generally to safeguarding a distributed system from any attack where an adversary seeks to obtain significantly more than its fair share of network resources. We formalize this guarantee in Theorem 1 of Section 1.2.

1.1 Our Model

Given space constraints, we summarize the key aspects of our model; a more in-depth discussion is provided in our full version [19].

Our system consists of virtual ***identifiers (IDs)***, and an attacker (or ***adversary***). Each ID is either good or bad. Each ***good*** ID follows our algorithm, and all ***bad*** IDs are controlled by a single adversary. We assume that all IDs know a ***hash function*** $\boldsymbol{h}$ that maps bit strings to real numbers in $[0, 1)$. We make the standard ***random oracle assumption*** about h [1, 5, 23, 27]. Let $\boldsymbol{\mu}$ denote the number of hash function queries that a good ID can make per unit time. Hence, we measure the computational power of a good ID in terms of μ.

Adversary. We assume that the adversary controls up to an $\boldsymbol{\alpha}$ ***fraction*** of the computational resources in the network, for α a constant that is a parameter of our algorithms. Our algorithms employ public key cryptography, and we make the usual cryptographic assumptions needed for this; however we do not assume a public key infrastructure (PKI). The adversary knows our algorithm, but does not know the private random bits of any good ID.

Communication. All communication among good IDs occurs through a broadcast primitive, denoted by **Diffuse**, which allows a good ID to send a value to all other good IDs within a known and bounded amount of time, despite the presence of an adversary. Time is discretized into ***rounds***. As a standard assumption, all IDs are assumed to be synchronized, but our algorithms can tolerate a small amount of skew. For simplicity, we initially assume that the time to diffuse a message is small in comparison to the time to solve computational puzzles. We pessimistically assume that the adversary can send messages to any ID at will, and that it can read the messages diffused by good IDs before sending its own.

Joins and Departures. The system is dynamic with IDs joining and departing over time (i.e., ***churn***), subject to the constraint that at most a constant fraction of the good IDs can join or depart in any round. Maintaining performance guarantees amidst churn is often challenging in decentralized systems. We pessimistically assume that all join and departure events are scheduled in a worst-case fashion by the adversary.

1.2 Main Result

We measure ***computational cost*** as the effort required to solve computational puzzles (see [19] for details), and we measure ***bandwidth cost*** as the number of calls to DIFFUSE. Let $\boldsymbol{T_C}$ and $\boldsymbol{T_B}$ denote the total computational and total bandwidth costs, respectively, incurred by the adversary. Let $\boldsymbol{g}_{\text{new}}$ denote the number of good IDs that have joined the system. The ***lifetime*** of the system is the time until at least $O(n_0^\gamma)$ for some fixed $\gamma \geq 1$, i.e., polynomially in n_0 join or leave events.

▶ **Theorem 1.** *Assume that the adversary holds at most an $\alpha = 1/6$-fraction of the total computational power of the network. Then, w.h.p. our algorithm (*CCOM *in Section 2) ensures the following properties hold over the lifetime of the system:*

1. *The fraction of bad IDs in the system is always less than $1/2$.*
2. *The cumulative computational cost to the good IDs is $O\left(T_C + g_{new}\right)$.*
3. *The cumulative bandwidth cost to the good IDs is $\tilde{O}\left(T_B + g_{new}\right)$.*

Note that the computational and bandwidth costs incurred by the good IDs grow slowly with the cost incurred by the adversary. Recalling our discussion at the beginning of this section, this is precisely the type of result we sought. When there is no attack on the system, the costs are low and solely a function of the number of good IDs; there is no excessive overhead. But as the adversary spends more to attack, the costs required to keep the system secure grow commensurately with the adversary's costs.

A variant of our algorithm can maintain a small committee that is known to all IDs, which enables scalable Byzantine agreement. This committee contains (1) a logarithmic number of IDs; and (2) less than a $1/2$ fraction of bad IDs.

Given space constraints, the details and analysis for our results are omitted here, but they can be found in [19, 18], along with a discussion of prior related work.

2 Our Algorithm

For simplicity, here we describe only a centralized algorithm; see [19] for the decentralized algorithm. We assume a ***server*** which can communicate directly with each ID in the system.

Algorithm 1: Commensurate Computation (CCom)

Input. The following sets are defined:
$\mathcal{S}_{\text{old}} \leftarrow$ set of IDs present at beginning of current epoch
$\mathcal{S}_{\text{t}} \leftarrow$ set of IDs present at time t

Execute the following steps for the lifetime of the system:

1. Upon joining, each ID v solves an entrance puzzle and sends the solution s_v to the server which adds v to $\mathcal{S}_{\text{t}}$ upon confirming the validity of s_v.
2. For any round t, if $|(\mathcal{S}_{\text{t}} \cup \mathcal{S}_{\text{old}}) - (\mathcal{S}_{\text{t}} \cap \mathcal{S}_{\text{old}})| \geq |\mathcal{S}_{\text{old}}|/3$, then the server:
   - Broadcasts a random string r to be used in the purge puzzle
   - $\mathcal{S}_{\text{old}} \leftarrow$ Set of IDs that returned valid solution
   - $\mathcal{S}_{\text{t}} \leftarrow \mathcal{S}_{\text{old}}$

2.1 Algorithm Overview

The centralized version of our algorithm *Commensurate Computation* (CCom) is given in Algorithm 1. Each ID that wishes to join the system and receive service must solve an ***entrance puzzle*** (Step 1), which is a computational puzzle that has 1 unit of computational cost and allows a node to enter the system. The only purpose of this puzzle is to force a computational cost on the adversary, even if this cost was incurred in the distant past.

The server tracks the membership in the system using the set $\boldsymbol{\mathcal{S}_{\text{t}}}$. Whenever an ID registers with the server, $\mathcal{S}_{\text{t}}$ is updated. Similarly, when a good ID informs the server that it is departing, $\mathcal{S}_{\text{t}}$ is also updated. However, bad IDs may not provide such a notification and, therefore, $\mathcal{S}_{\text{t}}$ is not necessarily accurate at all times.

At the start, the server knows the existing membership denoted by $\mathcal{S}_{\text{old}}$; assume $|\boldsymbol{\mathcal{S}_{\text{old}}}| = \boldsymbol{n_0}$ initially. At some point, $|(\mathcal{S}_{\text{t}} \cup \mathcal{S}_{\text{old}}) - (\mathcal{S}_{\text{t}} \cap \mathcal{S}_{\text{old}})| \geq |\mathcal{S}_{\text{old}}|/3$. When this occurs, it triggers the execution of Step 2 whereby all IDs are issued a ***purge puzzle*** of unit computational cost, and each ID must respond with a valid solution within 1 round; this is referred to as a ***purge***. The server issues a purge by broadcasting a random seed, $\boldsymbol{r}$, to all IDs; the value of r, and a node's public ID, must be incorporated into the solution to a purge puzzle.

Those IDs that fail to submit valid purge puzzle solutions during Step 2 are de-registered and permanently blacklisted — that is, they are removed from the system. Over the lifetime of the system, the execution of CCom is conceptually broken into sets of consecutive rounds called ***epochs***, which are delineated by the test in step 2. The process outlined above is repeated where now the server knows the existing membership *exactly*, $\mathcal{S}_{\text{old}}$, at the beginning of each epoch (thus, $\mathcal{S}_{\text{t}}$ is set to equal $\mathcal{S}_{\text{old}}$ at this point in the algorithm).

Initialization. Prior to the first epoch, the server issues purge puzzles to all IDs, and sets variables by running the steps under Step 2 of Algorithm 1. Thus, initially $\mathcal{S}_{\text{old}}$ contains less than a 1/3 fraction of bad IDs.

3 Empirical results

We simulate CCom to evaluate its performance against a well-known PoW scheme named SybilControl [24]. Details on the experimental setup are given in our full version [19].

To provide a fair comparison, we assume that the computational cost for solving a single PoW is 1 for both algorithms. Since both algorithms require that a new ID solve a puzzle to join the system, we refrain from measuring this computational cost. We let the fraction of

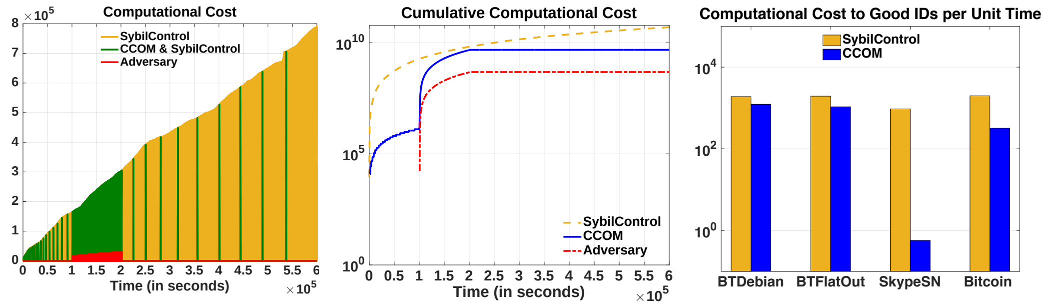

Figure 1 SybilControl and CCom during attack (left and middle) and without attack (right).

computational power of the adversary be the same for both algorithms.

The left and center subfigures of Figure 1 are generated using a real-world dataset for the Bitcoin Network [28, 29], where we simulate the following adversarial attack. Every 5 seconds, the adversary adds $\frac{n}{3}$ bad IDs to the network from time $\frac{t}{3}$ to $\frac{2t}{3}$, where n is the total number of IDs at the time and $t = 60,4970$ seconds (7 days) is the total execution time.

In the left subfigure, the green region represents the computational cost of the good IDs in SybilControl and CCom, the yellow region represents the cost of good IDs in SybilControl and the red region the cost of the adversary. Note that CCom incurs cost only at the start and end of each epoch. In contrast, SybilControl incurs a perpetual cost.

The center subfigure depicts the cumulative cost to the good IDs for SybilControl, CCom, and the adversary, and we make two observations. First, the cost of CCom is less than that of SybilControl. Second, CCom's cost is indeed a function of the cost paid by the adversary; in contrast, without a resource-competitive guarantee, SybilControl's cost grows at a significantly faster rate (note the logarithmic y-axis).

How is performance in the absence of an attack? The right subfigure summarizes the performance of the two algorithms on three different peer-to-peer (P2P) networks, namely BitTorrent, Skype, and Bitcoin [32, 17, 28, 29] when there is no adversary. We observe that CCom outperforms SybilControl in all four cases in terms of computational costs of the network. CCom outperforms SybilControl by 34.5% in BitTorrent Debian, by 45.6% in BitTorrent FlatOut, by 99.9% in Skype Supernodes and 83.9% in Bitcoin (again, note the logarithmic y-axis).

4 Conclusion and Future Work

We have described an algorithm to efficiently use PoW to reduce the fraction of bad IDs in open systems. Unlike previous work, our algorithm requires the good nodes to expend computational resources that grow only linearly with the computational resources expended by the adversary.

An interesting open problem is whether we can adapt our technique to ***secure multi-party computation (MPC)*** which involves designing an algorithm to compute the value of an n-ary function, f, over private inputs from n nodes $x_1, x_2, ..., x_n$, such that the nodes learn the value of $f(x_1, x_2, ..., x_n)$, but learn nothing more about the inputs than what can be inferred from this output of f. The problem is generally complicated by the assumption that an adversary controls a hidden subset of the nodes. We believe that our algorithms could be used to solve a dynamic version of secure MPC, while ensuring that the resource costs to the good nodes is commensurate with the resource costs to an adversary.

References

1 Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

2 Blockstack. Blockstack: The New Decentralized Internet, 2016. `https://blockstack.org/`.

3 Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *Proceedings of the IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 104–121, 2015.

4 Nikita Borisov. Computational Puzzles As Sybil Defenses. In *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, P2P '06, pages 171–176, 2006.

5 Ran Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. In *Proceedings of the Annual International Cryptology Conference*, pages 455–469. Springer, 1997.

6 Wu chang Feng, Ed Kaiser, Wu chi Feng, and Antoine Luu. The Design and Implementation of Network Puzzles. In *Proceedings of the Annual Joint Conference of IEEE Computer and Communications Societies (INFOCOM)*, pages 2372–2382, 2005.

7 Anthony Cuthbertson. Bitcoin Now Accepted by 100,000 Merchants Worldwide. February 2015. www.ibtimes.co.uk/bitcoin- now-accepted-by-100000-merchants-worldwide-1486613.

8 John Douceur. The Sybil Attack. In *Proceedings of the Second International Peer-to-Peer Symposium (IPTPS)*, pages 251–260, 2002.

9 Jamie Doward. City Banks Plan to Hoard Bitcoins to Help Them Pay Cyber Ransoms. October 2016. https://www.theguardian.com/technology/2016/oct/22/city-banks-plan-to-hoard-bitcoins-to-help-them-pay-cyber-ransoms.

10 Cynthia Dwork and Moni Naor. Pricing via Processing or Combatting Junk Mail. In *Proceedings of the* 12^{th} *Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147, 1993.

11 The Economist. The Magic of Mining. 2015. www.economist.com/news/business/21638124-minting-digital-currency-has-become-big-ruthlessly-competitive-business-magic.

12 The Economist. The Trust Machine. 2015. www.economist.com/news/leaders/21677198-technology-behind-bitcoin-could-transform-how-economy-works-trust-machine.

13 Ethereum. Ethereum: Blockchain App. Program, 2016. `https://www.ethereum.org/`.

14 Andy Extance. The Future of Cryptocurrencies: Bitcoin and Beyond. *Nature*, 526:21–23, 2015.

15 Sergey Gorbunov and Silvio Micali. Democoin: A Publicly Verifiable and Jointly Serviced Cryptocurrency. Cryptology ePrint Archive, Report 2015/521, 2015. `http://eprint.iacr.org/2015/521`.

16 Jeff Green, Joshua Juen, Omid Fatemieh, Ravinder Shankesi, Dong Jin, and Carl A. Gunter. Reconstructing Hash Reversal Based Proof of Work Schemes. In *Proceedings of the 4th USENIX Conference on Large-scale Exploits and Emergent Threats*, LEET'11, pages 10–10, 2011.

17 Saikat Guha and Neil Daswani. An experimental study of the skype peer-to-peer voip system. Technical report, Cornell University, 2005.

18 Diksha Gupta, Jared Saia, and Maxwell Young.

19 Diksha Gupta, Jared Saia, and Maxwell Young. Proof of Work Without All the Work. *CoRR*, abs/1708.01285, 2017. URL: `http://arxiv.org/abs/1708.01285`.

20 Robert Hackett. Can This 22-year-old Coder Out-Bitcoin Bitcoin? 2016. `http://fortune.com/ethereum-blockchain-vitalik-buterin/`.

21 Chain Incorporated. Chain: Enabling a New Medium for Assets, 2016. `https://chain.com/`.

22 Ed Kaiser and Wu chang Feng. Mod_kaPoW: Mitigating DoS with Transparent Proof-of-Work. In *Proceedings of the 2007 ACM CoNEXT Conference*, CoNEXT '07, pages 74:1–74:2, 2007.

23 Neal Koblitz and Alfred J Menezes. The Random Oracle Model: A Twenty-Year Retrospective. *Designs, Codes and Cryptography*, 77(2-3):587–610, 2015.

24 Frank Li, Prateek Mittal, Matthew Caesar, and Nikita Borisov. SybilControl: Practical Sybil Defense with Computational Puzzles. In *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing*, STC '12, pages 67–78, 2012.

25 Pablo Magro. What Greece can learn from bitcoin adoption in Latin America . July 2015. www.ibtimes.co.uk/what-greece-can-learn-bitcoin-adoption-latin-america-1511183.

26 Ivan Martinovic, Frank A. Zdarsky, Matthias Wilhelm, Christian Wegmann, and Jens B. Schmitt. Wireless Client Puzzles in IEEE 802.11 Networks: Security by Wireless. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 36–45, 2008.

27 Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. `http://bitcoin.org/bitcoin.pdf`.

28 Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In *Proceedings of the 13th IEEE International Conference on Advanced and Trusted Computing (ATC)*, July 2016.

29 Till Neudecker and Hannes Hartenstein. Could network information facilitate address clustering in bitcoin? In *International Conference on Financial Cryptography and Data Security*. springer, 2017.

30 Luke Parker. Blockchain Tech Startup Chain Inc Hits the Wall Street Motherload, 2015. http://bravenewcoin.com/news/blockchain-tech-startup-chain-inc-hits-the-wall-street-motherload/.

31 Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks. *ACM SIGCOMM Computer Communication Review*, 37(4):289–300, 2007.

32 Daniel Stutzbach and Reza Rejaie. Understanding Churn in Peer-to-peer Networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM. URL: `http://doi.acm.org/10.1145/1177080.1177105`, `doi:10.1145/1177080.1177105`.

33 Melanie Swan. *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc., 2015.

34 XiaoFeng Wang and Michael K. Reiter. Defending Against Denial-of-Service Attacks with Puzzle Auctions. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 78, 2003.

35 Brent Waters, Ari Juels, J. Alex Halderman, and Edward W. Felten. New Client Puzzle Outsourcing Techniques for DoS Resistance. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 246–256, 2004.