Blockchain and the Internet of Things: Improving the Proof of Work Consensus Algorithm

Andrew Levandoski, Jonathan Lobo, Christopher Corsi

***Abstract*—Internet of Things (IoT) security and privacy remain a major challenge due to the large scale and distributed nature of IoT networks. Approaches driven by blockchain (BC) offer decentralized security; however, they involve significant energy, delay, and computational overhead that may not be suitable for the most resource-constrained devices in an IoT network. In this paper, we discuss how the use of BCs can not only enhance the security of networks, but also enable new applications for IoT. Further, we examine the feasibility of the use of BCs for IoT networks, presenting simulation results to highlight the overheads in terms of energy consumption, CPU utilization, and memory writes introduced by proof-of-work (PoW), proof-of-stake (PoS), and efficient and equitable proof-of-work (ePoW) consensus algorithms. We show that under appropriate conditions, the ePoW consensus method enables the use of a BC for IoT networks as its associated overheads are outweighed by both gains in security and privacy and potential new applications.**

I. Introduction

IoT consists of devices that generate, process, exchange, and store security- and safety-critical data. As a result, many IoT devices are appealing targets for various forms of cyber-attacks aimed at obtaining and tampering with their data [1]. Further, the networked devices that constitute the IoT are constrained in their energy consumption and computational capability, devoting most of their available resources to core functionality. Thus, the task of affordably providing security and privacy using traditional methods can be challenging. In addition to their expense in terms of energy consumption and processing overhead, many security frameworks are centralized and not well-suited for IoT applications due to their difficulty of scale, many-to-one nature, and single point of failure [2]. In contrast, IoT demands lightweight, scalable, and distributed security. Devices can be configured to operate more safely and reliably with each other's conditions and requirements within a network using a private BC. Using a BC, an IoT network can be configured to perform both user authentication and mutual authentication between devices to generate and securely store operation records.

In addition to providing enhanced security, a BC-based IoT network enables trustless relationships between devices. Smart contracts – self-executing scripts that reside on the BC – can be integrated such that the network facilitates proper, distributed, and heavily automated workflows. However, the use of a BC within the context of IoT presents several challenges; most notable is the design of the block creation process, or mining, which resides at the core of a BC's functionality. While typically a computationally taxing procedure, mining must be designed such that it conforms to the constraints of IoT devices and does not compromise their core functionalities. Consequently, various consensus algorithms must be considered so that the proper method is selected based on the constraints of each particular IoT application.

The goal of this work is to provide a description of how BCs and smart contracts work within the context of an IoT network, to explore use cases for the union of these technologies, and to present an in-depth analysis of the implementation, performance, and feasibility of BC consensus algorithms for IoT networks. The rest of the paper is organized as follows: In Section II, we present related work. In Section III, we discuss what a BC is, how a BC network operates, consensus algorithms, and how smart contracts can be configured and automated. Section IV contains a discussion of smart contracts. Section V contains several use cases for the application of BCs to IoT networks. Section VI contains our simulation methodology and results. Finally, Section VII concludes the paper.

II. Reaching Consensus on a BC Network

A. Blockchains

A BC is a distributed data structure that is replicated and shared among members of a network. Introduced as a ledger for the cryptocurrency Bitcoin [3], a BC is a continuously growing list of records, called blocks, which are linked together. Nodes on the Bitcoin network, called miners, append validated, mutually agreed-upon transactions to the Bitcoin BC, forming an authoritative ledger of transactions that establishes who owns what [4].

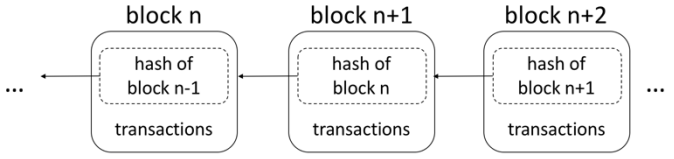

Fig. 1. Each block in the chain carries a list of transactions and a hash to the previous bloack. The exception to this is the first block of the chain which is common to all nodes in a network and has no parent.

Each BC is a log whose records of transactions are batched into blocks identified by unique hashes. The chain is formed with each block referencing the hash of the block before it (Fig. 1). Any node with access to the BC can read it to determine the global state of the data being exchanged on the network [5]. A BC network is a set of nodes that operate on the same BC via the copy that each one holds. While a node can can generally act as a gateway to the network for several users, we will assume for the sake of simplicity that each node transacts on behalf ofa

single user. These nodes form a peer-to-peer network where the following process is repeated:

1) Users interact with the BC using private and public keys [6]. Each user uses his private key to sign his own transactions which are then addressable on the network using his public key. Asymmetric cryptography offers authentication, integrity, and non-repudiation to the network. Each transaction is broadcasted by each node to its one-hop peers.
2) Neighboring peers validate incoming transactions prior to replaying them to the rest of the network. If a transaction is deemed to be invalid, it is discarded. If not, it is eventually proliferated across the entire network.
3) The transactions that have been collected and validated by the network within an agreed-upon time interval are ordered and packaged into a timestamped candidate block. This process is called mining. The mining node broadcasts the new block to the network. The choice of the mining node and the contents of the block depend on the consensus mechanism employed by the network. Consensus algorithms can be proof-of-work based (Section III-B), proof-of-stake based (Section III-C), or a modified version of several methods (Section III-D).
4) Nodes in the network verify that the proposed block contains valid transactions and references the correct previous block on the chain. If validated, the block is added to the chain and the transactions housed within it are applied to each node's world view. If not validated, the block is discarded.

Though the above is a high-level view of how a BC network functions, it is evident that with enough nodes in a network, there exists a platform through which a set of non-trusting writers can share a collection of data with no trusted intermediary [7]. The focus of this paper is how nodes can agree on transactions and the order in which they are listed on the newly-mined block. Without a means for consensus, individual copies of the BC will diverge, causing nodes to have different views of the world state and compromising the authoritative chronology of the chain. In the most simplistic scenario, all nodes could simply vote on the order of transactions for the next block, with the majority deciding the block. However, in an open network, a single entity could join with multiple identities to influence the network with multiple votes [8]. To circumvent this attack, BC network employ various schemes for the mining of new blocks, described in the following sections.

B. *Proof-of-Work*

The Bitcoin BC solves the problem of an easily compromised network by making mining a computationally expensive process; as a result, impersonating multiple identities on a network will not result in increased influence since the computing resources of any single actor are limited. This process, called proof-of-work (PoW), specifies that any node can have their assembled block mined on the network given that it can solve a complex task. Specifically, in the case of the Bitcoin BC, each node must determine the correct random number (nonce) in a block's header that will result in a SHA-256 hash [9] containing the number of leading zeroes that the network expects [10]. The nature of a one-way cryptographic hash function allows any node to easily verify that a given answer satisfies a network's requirements and adopt the associated block into its world view. It is possible that forks may still occur under this mechanism when multiples nodes successfully mine blocks nearly simultaneously. In this event, the fork is resolved by the next block since PoW dictates that nodes should adopt the fork that carries the greatest amount of work; i.e., whichever fork grows longer first will be used.

There are several difficulties associated with the use of a PoW scheme on a BC network. First, the consensus algorithm wastes energy by consuming hashing power to obtain one block compensation during the competitive mining process. Furthermore, energy is spent unnecessarily by the numerous nodes mining the same block concurrently. This inefficiency calls into question the feasibility of such a scheme in a resource-constrained IoT scenario. In addition, the PoW method is susceptible to concentrating compensation among nodes with greater hashing power.

C. *Proof-of-Stake*

A different method of validating transactions and achieving distributed consensus on a BC network is through a proof-of-stake (PoS) scheme. The gains of a PoS-based network come in the form of significantly less computation during the mining process. Instead, the creator of the next block is determined through a combination of random selection and wealth or age (i.e. a node's stake in the network). Aside from advantages in terms of energy-efficiency, PoS schemes offer a greater incentive strategy for member nodes since each node is seeking to guard its stake in the network through mining. In contrast, nodes can mine without possessing any stake under a PoW scheme and are thus only seeking to maximize their own gain.

However, several issues exist with PoS-based networks. Several plausible schemes exist for conspiring actors to sabotage a small enough PoS network by approving invalid blocks to the network and assuming a loss of currency in the process. Most notable, however, is the uneven distribution of wealth based on nodes' stakes. Nodes with a large stake in the network will be rewarded for their participation relative to their stake, granting the power to influence the network to a concentration of nodes. Several networks are working to solve such inequity [11].

D. *PoW Based on Equitable Chance and Energy-Saving*

Proposed by the BC-based IoT contract platform Hyundai Digital Asset Currency (HDAC), ePoW, an algorithm based on equitable chance and energy-saving, can significantly reduce the number of nodes actively participating in PoW. The motivation behind the scheme is to distribute equitable mining opportunities as well as to minimize energy waste from excessive hashing power spent during competitive mining.

The ePoW consensus algorithm reduces the mining monopoly using a block window concept (Fig. 2). By avoiding spontaneous mining attempts during the block window application period once mining is successful, ePoW reduces wasteful energy consumed in excessive hashing calculations. Even if a greedy node opts to neglect this prescribed mechanism and succeeds in mining a new block, its block will not be recognized as valid in the entire network. The block window size can be expressed in the form of a time function, $W_s = f(t)$, where $f(t)$ increases in proportion to time. As the window size gradually increases, so does the difficulty for nodes to monopolize the mining process. As a result, a more equitable distribution of mining is achieved in the network.

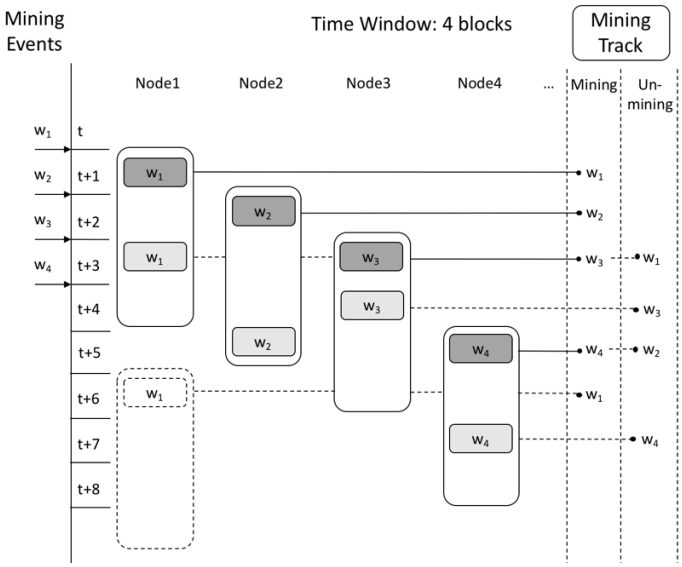

Fig. 2. The ePoW consensus algorithm.

III. Smart Contracts

A smart contract, introduced in 1994 by Nick Szabo, is defined as "a computerized transaction protocol that executes the terms of a contract" [12]. At the highest level, a smart contract is a contractual clause translated into code and embedded into hardware or software to enforce its terms so as to minimize the need for trusted intermediaries between transacting parties and the occurrence of malicious and accidental exceptions to its terms [13]. In the context of a BC-based IoT network, smart contracts are scripts stored on devices connected to the BC. Since the contracts reside on the BC, they have a unique address that can be referenced in a transaction when it is triggered; they can be executed independently and automatically executed in a prescribed manner on every node in the network.

A smart contract exhibits the following properties:

1) The contract has its own state and can take custody of assets on the BC [14].
2) The contract expresses business logic in code, e.g. "trade 1 unit of X for every 2 units of Y received".
3) The contract should describe all possible outcomes of the transaction that it is responsible for.
4) The relationship established between parties in the contract is data-driven.
5) The contract is triggered by messages and transactions sent to its address.
6) The contract is deterministic, i.e. the same input will always produce the same output. Without this requirement, a contract can execute on every node in the network and return different results for each node, preventing the network from reach consensus on the result. A BC-based platform should therefore employ only deterministic contracts.
7) The contract resides on the BC. Therefore, its implementation is visible to all network participants.
8) All network participants get a cryptographically verifiable trace of the contract's operations since all the contract's interactions occur via signed messages on the BC.

A BC-based network with smart contracts can support both transactions (asset transfers) and interactions (multi-step processes) between mutually distrustful counterparties since the transacting entities can inspect the contracts to identify outcomes before engaging in them, have certainty of the contract execution due to the distribution of the network, and have verifiability throughout the process since all interactions are digitally signed.

IV. Use Cases

A. Software Distribution

As IoT-based companies become more common and their devices proliferate, there will be an increasing need for them to distribute software in order to keep the IoT device ecosystem sustainable in terms of security and functionality [16]. The current centralized model of software distribution involves a high maintenance cost for manufacturers and a degree of cooperation on behalf of the consumers. These issues can be solved with a peer-to-peer model that can operate transparently and distribute data securely.

Consider a BC network one which all of a manufacturer's IoT devices reside. The manufacturer can deploy a smart contract that allows them to store the hash of its latest firmware update onto the network. The devices, having either shipped with the smart contract address or discovered it through another transaction, can query the contract to learn about and request the new firmware. While the first requests for a file will be serviced by the manufacturer's own node, later requests can be serviced by any node that the file has propagated to. As a result, software updates can occur automatically without any user interaction. Further, a device that joins the network after the manufacturer has ceased its participation can still retrieve necessary firmware updates and validate that it is receiving the correct files.

B. Sharing of Computational Resources

Fog computing extends the existing definition of Cloud computing to include the growing number of networked edge devices, including IoT devices, as well as smart clients among end users [19]. For example, a wearable device such as a smart watch might utilize a connected smartphone's computational resources to analyze the data that it collects as opposed to sending its data to a remote cloud. Fog computing thus creates an opportunity for users to save money and time on both the transportation of data and the cost of computation. A BC network can enable an economy of resource sharing across edge devices.

Large amounts of processing power and storage sit unused in the world's idle routers, tablets, phones, and sensors. Consider a BC network to which all of these devices are connected and on which a task can be distributed among the nodes. Depending on the latency constraints, price considerations, security restrictions, and the complexity of the task, smart contracts on the network can arrange to distribute

the task to nodes at prices already agreed upon. Since the accommodating resources would otherwise be left unused, such a sharing economy would be certain to bring greater efficiency to the existing landscape of computing services. In addition, latency-sensitive tasks could be more easily serviced on edge nodes than on the cloud, ensuring higher response times in time-sensitive applications.

C. Efficient Energy Distribution

The electricity sector, though still heavily based on massive, centralized power plants that generate power sent over long distances, is becoming increasingly distributed with smaller power generators and storage systems like solar panels and electric-vehicle batteries connected to the power grid [18]. Smaller components of this system can struggle to maximize their value due to the inefficiency of compensation for electricity production; however, a BC network with smart contracts could enable producers to receive compensation immediately.

With such a network, local nodes could trade energy with each other prior to selling electrons to the grid. Smart contracts on the network could prescribe rates based on surge times, power availability, and, in the case of wind- and solar-harvested energy, forecasted production. Benefits of an IoT-enabled network could extend to producers as well. For example, sensors and actuators located in windmills could be informed of a decrease in demand for electricity and turn off machinery that would otherwise experience unnecessary degrees of wear through extraneous operation. Further, power generated by renewable sources of energy could be easily distinguished from that generated by fossil fuels due to the transparency of the network.

D. Supply Chain Verification

In a supply chain example where a container leaves the manufacturer's site (point A), gets transported via railway to the neighboring port (point B), gets shipped to the destination port (point C), gets transported by truck to the distributor's facilities (point D), and finally reaches the retailer's site (point E) (Fig. 3), a BC-based IoT network can maintain a ledger verifiable by all parties involved in the movement of the container.

Historically, each stakeholder in this process maintains his own database to keep track of the asset and updates it based on inputs from other parties. In the proposed scenario, a BC network run across all of the IoT sensors and actuators along the supply chain would update with cryptographic verifiability, update automatically, and form an easily-auditable ledger of information.

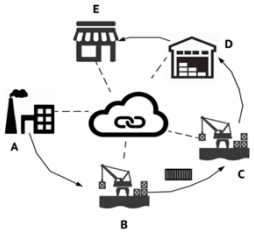

Fig. 3. An asset tracking example using smart contracts and IoT [16].

E. Intelligent Transportation Systems and Smart Transit

Intelligent transportation systems (ITS), a fast-growing sector involving IoT, cloud computing, and data-driven decisions, needs to consider many issues in its development including security, trust, and the social complexity of decisions between vehicles [17]. So that the ITS ecosystem maintains stability, safety, and effectiveness, there is a critical need to develop a secured, trustworthy, and decentralized architecture to facilitate the smooth and intermediator-free flow of data and assets among ITS entities. A BC network that connects sensors and actuators found in automobiles and traffic signals with users and key decision makers is an elegant solution for this application.

Next, consider a protocol called Transit that connects a city's cars, buses, taxis, etc. with users seeking transportation via internet-connected devices and maintains a distributed ledger that records all of its users' past trips, credit cards, favorite locations, etc. [15]. The standards for sending a Transit request onto the network would be entirely open and anyone who built an application to respond to the request is free to do so. For example, city governments could build Transit applications for their taxi drivers or public buses, ride-sharing companies can build applications for their employees, and bicycle-share collectives could build applications for their assets, all to field requests. Developers could create shared marketplace apps where all potential vehicles using Transit could compete for business in one place.

All of the phones and vehicle sensors connected to the Transit network could engage in smart contracts to determine a user's optimal pricing and time preference for a trip. Trip services could interact with each other to bid each other down to the point of optimal efficiency in terms of pricing, fuel consumption, distance traveled, requests pooled, and time spent in traffic for the user's request. This would involve the inclusion of vehicle, traffic, and gas station sensors on the BC network.

V. Evaluation and Analysis

A. Simulation Methodology

To determine whether or not a BC implementation is feasible in the context of an IoT network, we simulated the consensus algorithms previously discussed under several different scenarios on resource constrained devices. To approximate a miner with constrained resources, we used the Raspberry Pi 3 Model B, a device with 1 GB of LPDDR2 RAM, a 4x ARM Cortex-A54 1.2 GHz processor, and a 12GB flash hard drive. Using the Python progamming language, we implemented and ran three different consensus algorithms, PoW, ePoW, and PoS, all of which used SHA-256 for hash generation.

Each simulation included 1 – 3 physical nodes connected to the network and mining concurrently. To estimate the change in resource consumption as the length of a BC grows, the mining was simulated using 6 different difficulty levels, in whch the difficulty number corresponds to the number of leading zeros that must be present in an acceptable hash of a block. Each consensus algorithm was first simulated with only

a single node connected to the network to establish baseline resource consumption and then with three nodes mining to quantify any efficiencies of scale as well as any overhead introduced by network communication between nodes. Each of our simulations ran for 30 minutes with data collected at 20-second intervals. The data recorded includes CPU utilization, memory writes, blocks mined, and energy consumption in joules per block mined on the network. Power consumption was measured using a commercial Youthink Power Monitor. All figures reported are with respect to a single miner.

B. Results & Analysis

Table 1 presents numeric results from simulating each of the previously discussed consensus algorithms with either 1 or 3 miners on the network. The difficulty was fixed at 5 leading zeros for this comparison. After running all simulations, we found that PoS has by far the lowest energy consumption per block mined as well as the lowest CPU utilization. This is an expected result, as the intuition behind the PoS approach is to maintain consensus while avoiding any computationally heavy mining. Adding additional miners to the network does not affect the energy consumption or CPU utilization of an individual node in a PoS scheme since a node performs no work unless it is asked to verify a block (Fig. 4).

TABLE I
EVALUATION OF CONSENSUS ALGORITHMS

	Energy Consumed (J/Block)	**CPU Utilization**	**Memory Writes/Sec**
PoW	57.24	95%	0.044
PoW (3)	88.57	100%	0.044
ePoW	57.24	95%	0.044
ePoW (3)	25.21	98%	0.054
PoS	1	10%	.142
PoS (3)	1	10%	.142

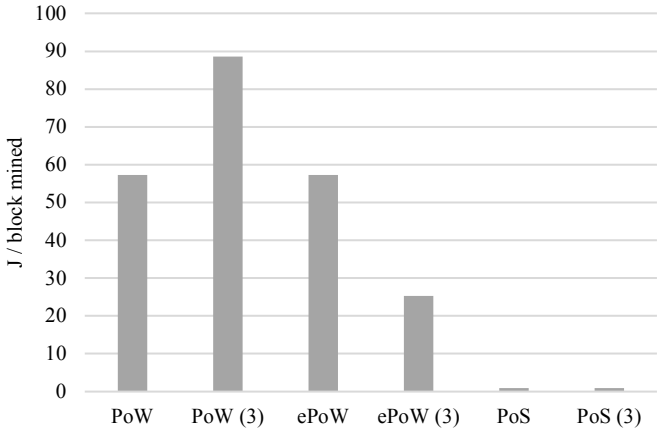

Fig. 4. Energy consumed by each method.

Both PoW and ePoW consume the same amount of energy when only a single miner is present. However, when the number of nodes within the network increases to 3, we note a significant drop in energy consumption for ePoW because the block window is preventing nodes from performing unnecessary mining. On the other hand, energy consumption actually *increases* when the nodes are following the PoW algorithm, likely due to the redundant mining operations. Both PoW and ePoW result in high CPU utilization due to the computationally

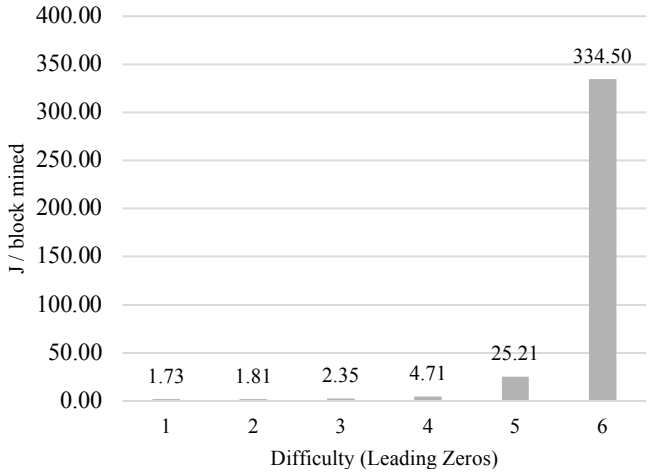

Fig. 5. Energy consumed by ePoW at difficulties 1-6.

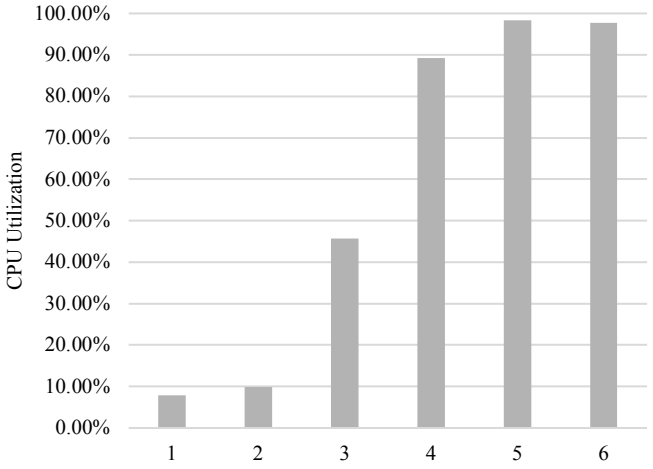

Fig. 6. ePoW CPU utilization at difficulties 1-6.

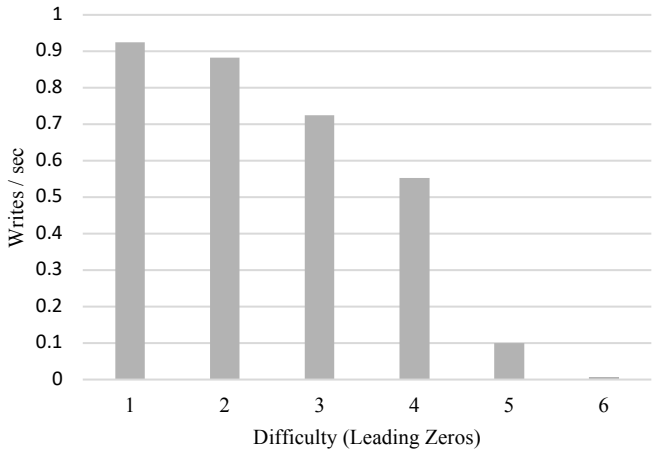

Fig. 7. ePoW writes per second at difficulties 1-6.

expensive mining process, and utilization increases further when there are more nodes on the network, presumably due to the added overhead of propagating blocks through the network.

Memory writes can partially be seen as a function of the rate at which blocks are being mined. A miner running PoW performs the same number of writes regardless of the number of nodes on the network because the total number of blocks mined is unaffected. Meanwhile, when ePoW runs with 3 miners, the average node writes more frequently because blocks are mined at a faster pace. The rate of memory writes for the PoS approach far exceeds either of the mining approaches due to the increased speed of verification. However, note that our simulation makes the assumption that there are infinite blocks available to be verified. In a real system, it is likely that the mining rate implied by our simulation of PoS would exceed the rate of actual block creation.

We also present data on energy consumption, CPU utilization, and writes per second for the ePoW algorithm with respect to hashing difficulty (Fig. 5 – 7). In these simulations, the number of nodes on the network was fixed at 3. We note that the search space becomes trivially small when the number of leading zeros is less than 4, leading to computation speeds that would likely exceed the speed at which blocks could be generated. On the other hand, mining on a constrained device becomes computationally impractical beyond a difficulty of 6.

We see that energy consumption per block mined appears to increase exponentially as the number of leading zeros in the required hash increases. Similarly, CPU utilization increases quickly along with difficulty before plateauing at near complete utilization after difficulty level 5. The trend of disk writes per second flows in the opposite direction and is indicative of the speed at which blocks are being mined. Unsurprisingly, as the difficulty of solving the hash increases, blocks are added to the chain at a slower rate and fewer writes occur.

VI. Conclusion

As we have demonstrated, a BC network powered by the IoT has the potential to power efficient, automated interactions between various parties in manner that minimizes the burden of trust while maintaining full verifiability. The combination of smart contracts in a BC with the rapidly expanding ecosystem of IoT devices can enable complex workflows that provide significant savings of energy, time, and money.

In this paper, we present a feasibility study of a BC network running on constrained devices, which as far as we know is the first of its kind. We implement three consensus algorithms, including the previously unimplemented ePoW, and evaluate them in terms of energy consumption, disk writes, and CPU utilization. We show that a BC network is feasible for a network of constrained IoT devices, particularly when the traditional PoW algorithm is augmented to promote equitable mining opportunities and mitigate superfluous mining.

In the future, we plan to simulate the consensus algorithms on a significantly larger network using secure communication protocols optimized for constrained devices so as to affirm the viability of a blockchain for the Internet of Things in a real-world system.

References

[1] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks,* vol. 76, pp. 146-164, 2015.

[2] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks,* vol. 57, no. 10, pp. 2266-2279, 2013.

[3] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[4] *Eris Industries Documentation – Blockchains,* accessed on Feb. 15, 2018. [Online]. Available: https://www.multichain.com/blog/2015/0 7/bitcoin-vs-blockchain-debate.

[5] *Eris Industries Documentation – Blockchains,* accessed Feb. 15, 2018. [Online]. Available: https://docs.erisindustries.com/explainers/ blockchains/.

[6] (2005). *Understanding Public Key Cryptography.* [Online]. Available: https://technet.microsoft.com/en-us/library/aa998077(v=exchg.65).aspx.

[7] G. Greenspace. (2015). *Avoiding the Pointless Blockchain Project.* [Online]. Available: https://www.multichain.com/blog/2015/11/ avoiding-pointless-blockchain-project/.

[8] J.R. Douceur, "The Sybil attack," in *Peer-to-Peer Systems* (Lecture Notes in Computer Science). Berlin, Germany: Springer, Mar. 2002, pp. 251-260. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-45748-8_24.

[9] (Aug. 1, 2002). *Announcing the Secure Hash Standard.* [Online]. Available: https://csrc.nist.gov/publications/fips/fips180-2.pdf.

[10] A.M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Crypto-currencies,* 1st ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2014.

[11] Prisco, Giulio (Nov 29, 2017). "The Ethereum Killer Is Ethereum 2.0: Vitalik Buterin's Roadmap". *Bitcoin Magazine*. Retrieved Jan 19, 2018.

[12] N. Szabo. (1994). *Smart Contracts.* [Online]. Available: http://szabo.best.vwh.net/smart.contracts.html.

[13] N. Szabo. (1997). *The Idea of Smart Contracts.* [Online]. Available: http://szabo.best.vwh.net./smart_contracts_idea.html.

[14] R.G. Brown. (2015). *A Simple Model for Smart Contracts.* [Online]. Available: http://gendal.me/2015/02/10/a-simple-model-for-smart-contracts.

[15] S. Johnson. (2018). *Beyond the Bitcoin Bubble.* [Online]. Available: https://www.nytimes.com/2018/01/16/magazine/beyond-the-bitcoin-bubble.html.

[16] K. Christidis and M. Devetsikiotis. "Blockchains and Smart Contracs for the Internet of Things." *Special Section on the Plethora of Research in Internet of Things,* col. 4, pp 2292-2303, 2016.

[17] Y. Yuan and F. Wang. "Towards Blockchain-based Intelligent Transportation Systems." *2016 IEEE International Conference on Intelligent Transportation Systems*.

[18] M. Orcutt. (2017). *How Blockchain Could Give Us a Smarter Energy Grid.* [Online]. Available: https://www.technologyreview.com/s/609077 /how-blockchain-could-give-us-a-smarter-energy-grid/.

[19] L. Zheng and C Joe-Wong. (2017). *Fogonomics: Pricing and Incentivizing Fog Computing.* [Online]. Available: https://www.open-fogconsortium.org/fogonomics-pricing-and-incentivi-zing-fog-computing/.