A Template for Alternative Proof of Work for Cryptocurrencies

Sajedul Talukder
Southern Illinois University
Carbondale, IL, USA
sajedul.talukder@siu.edu

Riley Vaughn
Edinboro University
Edinboro, PA, USA
rv161780@scots.edinboro.edu

***Abstract*—Many popular cryptocurrencies, such as Bitcoin, form consensus through a method known as Proof of Work. Problematically, current implementations of Proof of Work require immense amounts of energy consumption, where a majority of this energy is spent solely on securing consensus. Our focus is not to directly decrease energy consumption, but to allow for more useful and pragmatic computation to come from Proof of Work, such that energy is saved by not running these computational tasks separately. In this paper, we create a template for Proof of Work protocols, such that if followed, can guarantee similar security assurances as to the Proof of Work present in Bitcoin. Secondarily, we also develop "useful" prototypes based on this template.**

***Keywords*—Cryptocurrencies, PoW, Oracle, Factorization, Hash.**

I. Introduction

A cryptocurrency, commonly known as "crypto" is a digital currency that uses an online ledger with powerful cryptography. Trading for gains is a major part of the interest in these unregulated currencies, with their usage extending from social networks [1]–[5] to e-governance [6]–[8], from digital automation [9], [10] and cybersecurity [11], [12] to cellular networks [13]. Distributed systems such as cryptocurrencies, use consensus algorithms in order to achieve agreement on data. The type of algorithm used by Bitcoin and many other cryptocurrencies for this purpose implements a concept known as Proof of Work (PoW) [14]. In general, a PoW algorithm requires that a large amount of processing power be used to solve an easily verifiable problem. Unfortunately, current implementations of PoW require immense amounts of energy, such as the Bitcoin Network, which has an estimated annual power consumption of 67.38 TWh [15], see Figure 1. Bitcoin's PoW algorithm adopts the HashCash cost function [16], which operates by brute-forcing the SHA-256 hash of a block, while slightly modifying it each iteration until the block's hash has a predetermined number of leading zeroes [14]. Naturally, as SHA-256 is conjectured to have uniformly distributed output [17], most hashes do not have the correct number of leading zeroes, and are consequently discarded. The intent of this project is to decrease PoW's energy waste. Our approach is not to directly decrease energy consumption, but rather to make less waste from such consumption.

To create less waste from a PoW, the outputs ought to be useful. Attempts have been made previously to create such

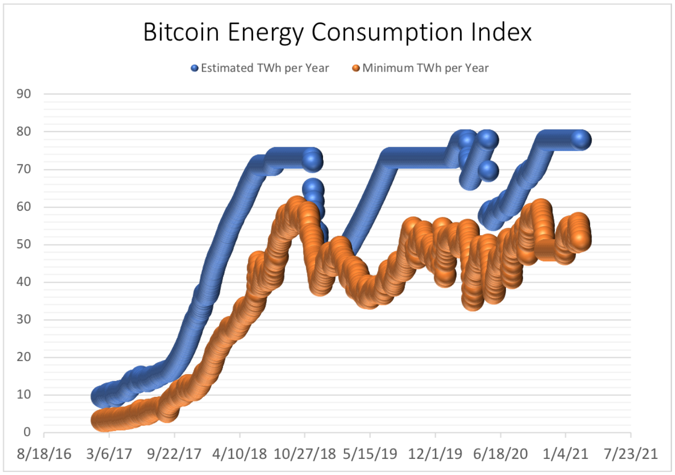

Fig. 1. Bitcoin Power Consumption

systems, but how can we determine if they are as secure as Bitcoin's PoW? Thus, the objective of this paper is to create a template for useful Proofs of Work (uPoW), such that if followed, can guarantee that a protocol has the same security assurances as Bitcoin's PoW. We make no guarantees as to the security of Bitcoin's PoW, but rather we guarantee that a protocol will have similar security to it. We only discuss the security of the PoW and have no interest in other cryptocurrency security concerns such as network specifications. We first review the literature on cost function properties and alternative consensus mechanisms, then flesh out the specific parameters of a PoW. We then create a template for alternative PoW and develop prototype designs for alternative PoWs based on our template. To the best of our knowledge, this paper is the first attempt to create a template for alternative PoW.

Our Contributions. This paper presents the following contributions:

- **Survey Consensus Protocols**. Review the literature on cost function properties and alternative consensus mechanisms, then identify the specific parameters of a PoW.
- **Template for Alternative PoW**. Create a template for alternative PoW and develop prototype designs for alternative PoWs based on our template.
- **List Sort PoW**. Develop a rudimentary prototype "List

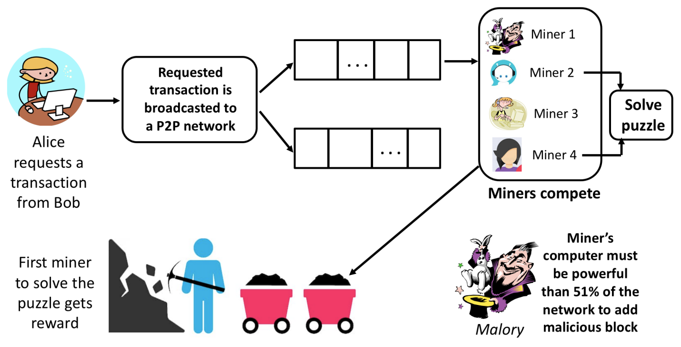
Alice requests a transaction from Bob
Requested transaction is broadcasted to a P2P network
Miner 1
Miner 2
Miner 3
Miner 4
Solve puzzle
Miners compete
First miner to solve the puzzle gets reward
Miner's computer must be powerful than 51% of the network to add malicious block
Malory

Fig. 2. Proof of Work Mechanism

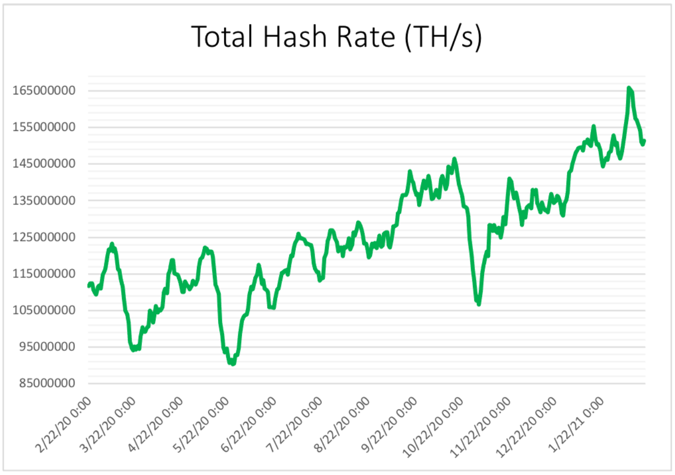

Fig. 3. Bitcoin estimated number of terahashes per second over 24 hours

Sort PoW" based on our PoW template. Its external usefulness is similar to that of Bitcoin's PoW.

- **Integer factorization PoW**. Propose another prototype called "Integer Factorization PoW" that creates a dictionary of a number and its factors.

The rest of the paper is organized as follows. Section II describes the background of the work. Section III presents the literature review. Section IV discusses the proof of work parameters. Section V presents a template for proof of work. Section VI discusses the limitations of our paper. Finally, Section VII concludes the paper with a highlight on the scope of future work.

II. Background

A type of public ledger known as a "blockchain." is used for cryptocurrencies such as Bitcoin. This ledger contains a log of all bitcoin transactions, organized in sequential "blocks," so that no user is able to invest any of their assets twice. The ledger is public or "distributed" to avoid tampering; a changed version will be easily discarded by all users.

In fact, the way that users detect tampering is by hashes, long strings of numbers that function as evidence of work. For a modern machine, creating just any hash for a series of bitcoin transactions will be easy, but the bitcoin network sets a certain amount of "work," in order to transform the operation into "difficulty." This setting is modified so that a new block is "mined" - added to the blockchain through generating a correct hash-approximately every 10 minutes. By setting a "target" for the hash, setting difficulty is achieved: the lower the target, the smaller the set of possible hashes, and the harder it is to produce one. This implies, in practice, a hash that begins with a long string of zeros. If a user modified one bitcoin transaction number by 0.0001, the subsequent hash will be unrecognizable, and the scam will be refused by the network.

Miners modify the input by adding an integer, called a nonce, to ensure that they produce a hash below the target. It is transmitted to the network until a correct hash is detected, and the block is added to the blockchain. Mining is a competitive mechanism in which newly generated bitcoins are awarded, see Figure 2. Proof of work renders it exceedingly impossible to change any part of the blockchain, as such a modification would entail re-mining all costly subsequent blocks.

III. Related Work

A. Cost Function Properties

In Adam Back's Hashcash paper, he refers to a CPU cost function, as a function that proves computational work has been done [16]. In general, cost function consists of six properties: efficiency of verifiability, bounds of cost, level of interactivity, public auditability, existence of a trapdoor, and parallelizability [16].

Verifying Bitcoin's PoW is as simple as executing a single SHA-256 hash [14]. Since SHA-256 is not bijective [17], the probability that the hash of a block will never produce a result with the proper amount of leading zeros will asymptotically approach zero, and such Bitcoins's PoW has unbounded probabilistic cost. Since problem generation is local, the system is non-interactive. Due to the decentralized nature of the system, it is both publicly auditable and trapdoor free. Bitcoin's PoW is also parralizable. And thus, to be consistent with Bitcoin's PoW, a PoW ought to be efficiently verifiable, have unbounded probabilistic cost, be publicly auditable, be trapdoor-free, and be parallelizable. Figure 3 shows Bitcoin's estimated number of terahashes per second over 24 hours from Feb 2020 to Feb 2021 which shows a similar trend to Bitcoin's power consumption.

B. Proof-of-Work Mechanisms

Blockchain-based cryptocurrencies have shown how historically centralized structures, such as currencies, can be applied security and privacy in a decentralized manner. [18]–[21]. King [22] proposed a new type of proof-of-work based on searching for prime numbers in peer-to-peer cryptocurrency designs. Chepurnoy et al. [23] proposed TwinsCoin, the first cryptocurrency based on a provably secure and scalable public blockchain design using both proof-of-work and proof-of-stake mechanisms.

Sleiman et al. [24] introduced an approach for data insertion on a Proof-of-Work cryptocurrency system. Bentov et al. [25] proposed a new protocol for a cryptocurrency, that builds upon the Bitcoin protocol by combining its Proof of Work component with a Proof of Stake type of system. Vukolić [26] showed poor performance scalability of early PoW blockchains no longer makes sense and contrasted PoW-based blockchains to those based on BFT state machine replication, focusing on their scalability limits.

Wustrow [27] presented a cryptocurrency with a malicious proof-of-work that allows miners to prove that they have contributed to a distributed denial of service attack against specific target servers. Consequently, Kiayias [28] introduced cryptographic primitive, Non-Interactive Proofs of Proof-of-Work (NIPoPoWs) that puts forth the first side chains construction that allows communication between proof-of-work blockchains without trusted intermediaries.

C. Alternative Consensus Protocols

There have been many previous attempts to create cryptocurrency consensus protocols which use different cost functions than Bitcoin. Some of these protocols continue to use computational work as the cost function's resource, while others have derived alternatives.

In Sunny King's 2013 paper, he introduces a PoW alternative to HashCash [17]. By discovering Cunningham chains and bi-twin chains, Primecoin is able to have mathematical importance while also securing the network. Primecoin is one of the first examples of a PoW that has significance, albeit limited, outside of securing a network.

PieceWork is a protocol designed to solve the problems of mining pool centralization and PoW usefulness. The protocol does so by splitting the PoW into an inner puzzle, and an outer puzzle, in which the inner puzzle is outsourceable and the outer puzzle is not [29]. Inner puzzles can also be useful, and solve problems outside of blockchain security, yet the paper does not delve into great detail regarding the specifics of implementing useful inner puzzles.

Ball et al. [30] have developed PoWs based on the Orthogonal Vectors, 3SUM, and All-Pairs Shortest Path problems. If Bitcoin's SHA-256 based PoW were to be broken, this would have little mathematical significance, as SHA-256 has only ever been heuristically shown to have uniform distribution [17], yet breaking a PoW based on any of the aforementioned problems would have far-reaching mathematical implications.

It was previously conjectured that in order to have a PoW that could be efficiently verified, it must also be parallelizable [16], yet both Momentum[8] and Cuckoo Cycle [31] are protocols which aim to maintain efficient verifiability but decrease parallelizability. They do so because the Bitcoin network has been overtaken by graphics card and ASIC miners [8,9], which consequently increases centralization and decreases the security of the system.

Gridcoin [32] and CureCoin [33] are cryptocurrencies that intend to use computational power to solve important medical, scientific, and mathematical problems. Rather than securing the network with the same computation as is used to solve these problems, both systems secure the network using Proof of Stake, and use the energy and computational power used from this to solve the aforementioned problems.

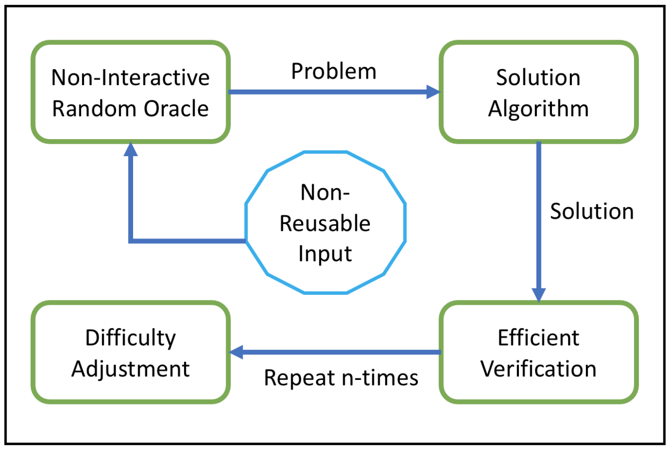

Fig. 4. Generalized PoW Template

IV. Proof of Work Parameters

A. Random Oracle

An Oracle is a third party from which one can retrieve data. In practice, an oracle can range from a simple, locally run function, to a central database, to a decentralized peer-to-peer network. In the context of a PoW, an Oracle generates the problems to be solved. The image of SHA-256 is conjectured to be uniformly distributed [17], which means that for any arbitrary input there is an equal chance of receiving any response in the function's output domain. The Oracle that Bitcoin's PoW employs is referred to as a Random Oracle for this reason. If an Oracle did not produce problems with random solutions, then an algorithm with runtime faster than brute force could be used by the adversary to outperform the honest nodes. Therefore, it is necessary that the Oracle be a Random Oracle.

B. Interactivity

In HashCash [16], Back characterizes a cost function as being either interactive or non-interactive, where interaction occurs when a client interacts with an external source for information. When it comes to a PoW, an Oracle is interactive if problems are generated externally and non-interactive if generated locally. For many useful PoW algorithms, a non-interactive Random Oracle would not be sufficient. For example, in order to use the folding of proteins as a PoW with security guarantees similar to Bitcoin's, it would be necessary to randomly generate the proteins being folded. The non-interactive way to do this would be to create a function which outputs randomly generated proteins to be folded. Without intimate knowledge of this specific research area, we conjecture that folding such proteins would have limited usefulness In many situations, a truly useful Random Oracle needs to be interactive. Thus, it is necessary to show that an interactive Random Oracle can exist and be just as

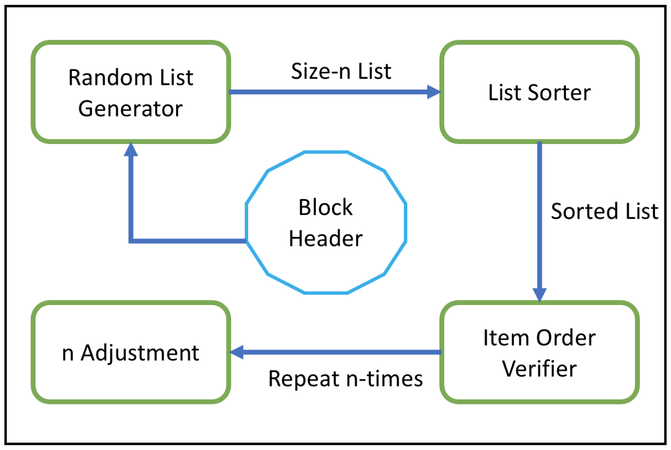

Fig. 5. List Sort PoW

secure as a non-interactive Random Oracle. This is an open question we hope to address at a later date but for now we will assume the Random Oracle must be non-interactive.

C. *Efficiency of Verifiability*

It is essential that solutions to the problems generated by the Random Oracle be efficiently verifiable. To be efficiently verifiable is to use significantly fewer computational resources than were used to find the solution initially [16]. Efficient verifiability is important because verification is used to audit whether or not work has been done. If Bitcoin was inefficiently verifiable, to see if one had received money or to view any other transaction on the blockchain, would require a similar amount of computational resources as was spent by the original block miner. These heavy computational loads would make the technology unusable, and thus it is ideal that the solutions to the PoW are efficiently verifiable.

D. *Auditability and Trapdoors*

Because of the decentralized nature of Bitcoin, it's PoW is both publicly auditable and trapdoor free. A trapdoor in the context of a cost function, is the ability for a party to circumvent the normally required work in order to create a PoW without actually doing the work [16]. In a trustless system such as Bitcoin, a trapdoor does not make sense. If any alternative PoW had a trapdoor, it would not be decentralized and would thus not have similar security to that of Bitcoin's. In order to be publicly auditable, a PoW must be able to be easily verified by an arbitrary third party without the use of a trapdoor [16]. As previously shown, an alternative PoW ought to be efficiently verifiable and cannot have a trapdoor.

E. *Bounds of Cost*

There are three ways an Oracle function can be bounded: fixed cost, bounded probabilistic cost, and unbounded probabilistic cost. Bitcoin's Oracle problems have unbounded probabilistic cost, for the chance of generating a hash without the proper leading zeroes asymptotically approaches zero as attempts increase. Yet, alternative PoW problems must not necessarily have unbounded probabilistic cost in order to be as secure. For example, the probability that a coin lands on heads, is equal to that of tails. As explained by Back [16], the procedure of flipping a coin has unbounded probabilistic cost. A PoW based on coin flips might require the actor to flip heads in order to add a block to the chain. After an arbitrary amount of (coin-flips) w, the average amount of work done per block would be 2w. A fixed cost system could replicate this result by allowing a worker to only add a block once they have flipped the coin twice, resulting in a 2w cost per block. Similarly, a bounded probabilistic cost system could remove the option of tails from the pool of options if flipped first. This would result in a probability of 1.5w per block, so we would weight the system by allowing 3 blocks to be created every 4 times this work was done (3 blocks = 6w), which would average to 2w per block. A potential issue of using a fixed or unbounded probabilistic cost problem set is the efficiency of verification. It might be difficult to consistently verify a problem faster than it can be solved if the problem can be solved in fixed time.

F. *Parallelizability*

A PoW is parallelizable if it is faster to compute multiple of the Random Oracles problems concurrently, than to compute them sequentially. Originally, non-parallelizable cost functions were orders of magnitudes slower to verify than their parallelizable counterparts [16]. In Momentum [34], Larimer discusses a memory-bound PoW which is efficiently verifiable. Similarly, Tromp [31] discusses a generalized version of Larimer's method. While these cost functions are still parallelizable, they are not bound by computation alone, but also by memory. While memory-bound PoW does not completely remove parallelization, it does make the solution algorithm more difficult to run in parallel. Nonetheless, the ability to parallelize the work does not add to the security of Bitcoin's PoW. Therefore, parallelizability is not necessary for an alternative PoW to have similar security assurances to those of Bitcoin's.

G. *Adjustable Difficulty*

The difficulty of Bitcoin's PoW is adjusted every 2016 blocks, with a new block to be mined approximately every 10 minutes. [14] This is done in order to maintain consistent levels of inflation and security as resources enter and exit the network. In order to adjust the difficulty of the PoW, Bitcoin changes the number of leading zeroes necessary on a hash. One obstacle alternative PoWs might face is linear difficulty adjustment. Primecoin uses the remainder of the Fermat test to construct a relatively linear curve [22]. Cuckoo Cycle adjusts difficulty by changing the ratio of m/n in the bipartite graph. In order for a PoW to be similarly as secure as Bitcoin's PoW, the Random Oracle must have linear adjustable difficulty.

H. *Non-Reusability*

It is important that computational resources used to mine one block cannot be re-used to mine another. In order to be non-reusable, Bitcoin uses the current block as input into the PoW function. Some useful PoWs might not use the current block as input when requesting a problem from the Random Oracle, so it is important that they address this issue with other schemes.

V. A Template for Proof of Work

A. List Sort PoW

The List Sort PoW is a rudimentary prototype based on our PoW template, see Figure 5. Its external usefulness is similar to that of Bitcoin's PoW.

Algorithm 1: List Sort PoW

```
Data: Hash of the block header
Result: Publicly auditable PoW
initialization;
while True do
    take the hash of the block header as input;
    generate a size n list of random elements;
    for $k \leftarrow 1$ to $n$ do
        recursively hash the previous output;
        append each output to the final list;
    end
end
sort the list;
verify if every item in the list is in order;
return
```

- The Oracle takes the hash of the block header as input and produces a size n list of random elements as output. This Oracle is implemented by recursively hashing the previous output n times, appending each output to the final list.
- The worker sorts the list with any efficient sorting algorithm. To verify, an actor checks if every item in the list is in order. Verification O(n) is more efficient than sorting the list O(nlogn).
- Since the Oracle is truly random, there are no trapdoors, and since every actor has access, the PoW is publicly auditable.
- n is linearly adjustable by desired difficulty.
- Since every block header is unique, the solution to any Oracle problem cannot be reused.
- To achieve non-interactivity, the Oracle is implemented as a local function.

In order to increase the usefulness of List Sort PoW, interactivity would be necessary. If the domain of all possible lists to be chosen by the Random Oracle were externally provided, this PoW could operate as a list sorting service. Sadly, since the security of interactivity has not been shown, the protocol must remain non-interactive, in order to assure similar security to Bitcoin's PoW.

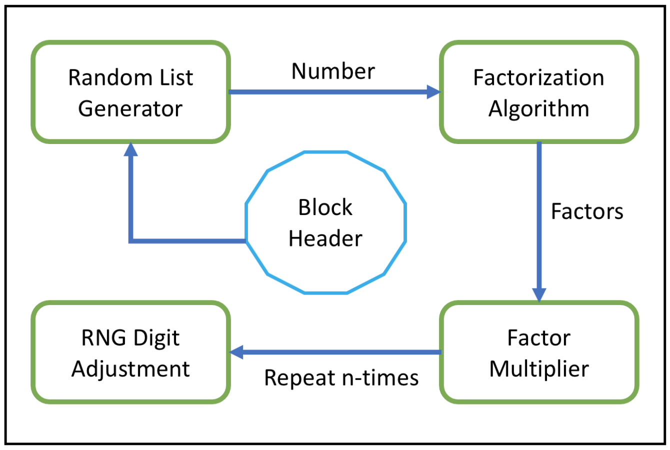

Fig. 6. Integer Factorization PoW

B. Integer factorization PoW

The Integer Factorization PoW is slightly useful because it creates a dictionary of a number and its factors, but is still not extremely useful because all numbers are randomly generated, not actor specified. Figure 6 shows the integer factorization PoW.

Algorithm 2: Integer factorization PoW

```
Data: Hash of the block header
Result: Publicly auditable PoW
initialization;
while True do
    take the hash of the block header as input;
    produce an integer in hexadecimal format;
    for $k \leftarrow 1$ to $n$ do
        adjust number of digits from RNG;
    end
end
use factoring & multiply the factors;
verify if the product equals the original hash;
return
```

- The Oracle takes the hash of the block header as input and produces an integer in hexadecimal format.
- The worker uses any efficient factoring algorithm. To verify, the actor multiplies the factors and determines if the product equals the original hash. Verifying is more efficient than calculating for large numbers.
- Because the Oracle is truly random, there are no trapdoors, and since every actor has access, the PoW is publicly auditable.
- To adjust difficulty, the Oracle should remove digits from the front of the hash.
- Since every block header is unique, the solution to any Oracle problem cannot be reused.

- To achieve non-interactivity, the Oracle is implemented as a local function.

VI. Limitations

The main limitation for useful PoW is the inability to use an interactive Oracle. If in the future, interactive Random Oracles are shown to exist with similar security to their non-interactive counterparts, PoWs will become truly useful. This is because instead of solving randomly generated problems from a given domain, the protocol will be able to solve specific problems actors want to be solved. While this PoW could prove moderately useful by providing a dictionary of numbers and their factors, the likelihood that an external actor could find the factorization of the number they want is minimal. If the Oracle was interactive and produced a number from a domain of numbers the actor required be factorized, this PoW could be truly useful.

VII. Conclusions

Present Proof of Work applications require large amounts of consumption of energy, where the majority of this energy is spent purely on creating consensus. Our goal is not to specifically reduce energy consumption, but to make it easier for Proof of Work to deliver more realistic and useful computation, in order to conserve resources by not doing these computing activities separately. In our paper, we develop a template for proof of work protocols in such a way that a protocol with identical security assurances found in Bitcoin can be ensured if followed. Additionally, we develop "useful" prototypes based on this template.

References

[1] B. Guidi, "When blockchain meets online social networks," *Pervasive and Mobile Computing*, vol. 62, p. 101131, 2020.

[2] S. Talukder and B. Carbunar, "A study of friend abuse perception in facebook," *Transactions on Social Computing*, vol. 1, no. 1, 2020.

[3] M. U. Rahman, B. Guidi, and F. Baiardi, "Blockchain-based access control management for decentralized online social networks," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 41–54, 2020.

[4] S. Talukder and B. Carbunar, "AbuSniff: Automatic Detection and Defenses Against Abusive Facebook Friends," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[5] S. K. Talukder, "Detection and prevention of abuse in online social networks," *FIU Electronic Theses and Dissertations. 4026*, 2019. [Online]. Available: https://digitalcommons.fiu.edu/etd/4026

[6] S. Naphade, H. Dubbewar, M. Patil, and S. Tambave, "Ethereum blockchain based e-governance system," 2019.

[7] S. K. Talukder, M. I. I. Sakib, and M. M. Rahman, "Model for e-government in bangladesh: A unique id based approach," in *2014 International Conference on Informatics, Electronics Vision (ICIEV)*, May 2014, pp. 1–6.

[8] M. Pilkington, R. Crudu, and L. G. Grant, "Blockchain and bitcoin as a way to lift a country out of poverty-tourism 2.0 and e-governance in the republic of moldova," *International Journal of Internet Technology and Secured Transactions*, vol. 7, no. 2, pp. 115–143, 2017.

[9] S. Talukder, M. I. I. Sakib, Z. R. Talukder, U. Das, A. Saha, and N. S. N. Bayev, "Usensewer: Ultrasonic sensor and gsm-arduino based automated sewerage management," in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*. IEEE, 2017, pp. 12–17.

[10] L. Thomas, C. Long, P. Burnap, J. Wu, and N. Jenkins, "Automation of the supplier role in the gb power system using blockchain-based smart contracts," *CIRED-Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 2619–2623, 2017.

[11] S. Talukder and Z. Talukder, "A survey on malware detection and analysis tools," *International Journal of Network Security & Its Applications*, vol. 12, no. 2, 2020.

[12] J. Moradi, H. Shahinzadeh, H. Nafisi, G. B. Gharehpetian, and M. Shaneh, "Blockchain, a sustainable solution for cybersecurity using cryptocurrency for financial transactions in smart grids," in *2019 24th Electrical Power Distribution Conference (EPDC)*. IEEE, 2019, pp. 47–53.

[13] S. Talukder, I. I. Sakib, F. Hossen, Z. R. Talukder, and S. Hossain, "Attacks and defenses in mobile ip: Modeling with stochastic game petri net," in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 2017, pp. 18–23.

[14] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, vol. 4, 2008.

[15] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," 2014.

[16] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[17] M. Green, "What is the random oracle model and why should you care, parts i, ii, iii and iv," 2011.

[18] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–34, 2019.

[19] S. Talukder and B. Carbunar, "When friend becomes abuser: Evidence of friend abuse in facebook," in *Proceedings of the 9th ACM Conference on Web Science*, ser. WebSci '17. New York, NY, USA: ACM, June 2017. [Online]. Available: http://doi.acm.org/10.1145/3091478.3098869

[20] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 439–457.

[21] S. Talukder, "AbuSniff: An automated social network abuse detection system," *J. Comput. Sci. Coll.*, vol. 35, no. 3, p. 209–210, Oct. 2019.

[22] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," *July 7th*, vol. 1, no. 6, 2013.

[23] A. Chepurnoy, T. Duong, L. Fan, and H.-S. Zhou, "Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake." *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 232, 2017.

[24] M. D. Sleiman, A. P. Lauf, and R. Yampolskiy, "Bitcoin message: Data insertion on a proof-of-work cryptocurrency system," in *2015 International Conference on Cyberworlds (CW)*. IEEE, 2015, pp. 332–336.

[25] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[26] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*. Springer, 2015, pp. 112–125.

[27] E. Wustrow and B. VanderSloot, "Ddoscoin: Cryptocurrency with a malicious proof-of-work," in *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.

[28] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 21–34.

[29] P. Daian, I. Eyal, A. Juels, and E. G. Sirer, "(short paper) piecework: Generalized outsourcing control for proofs of work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 182–190.

[30] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of work from worst-case assumptions," in *Annual International Cryptology Conference*. Springer, 2018, pp. 789–819.

[31] J. Tromp, "Cuckoo cycle: a memory bound graph-theoretic proof-of-work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 49–62.

[32] Gridcoin, "Gridcoin white paper," Gridcoin, https://gridcoin.us/assets/img/whitepaper.pdf, 2020.

[33] Curecoin, "Curecoin white paper," Curecoin, https://curecoin.net/white-paper/, 2019.

[34] D. Larimer, "Momentum–a memory-hard proof-of-work via finding birthday collisions," Tech. Rep., Oct. 2013.[Online]. Available: http://invictus-innovations. com ..., Tech. Rep., 2014.