# A Cost-efficient Protocol for Open Blockchains

Chunlei Li
Department of Informatics,
University of Bergen, Bergen, Norway
Chunlei.Li@uib.no

Chunming Rong and Martin Gilje Jaatun
Dept. of Electrical Engineering and Computer Science,
University of Stavanger, Stavanger, Norway
{Chunming.Rong,Martin.G.Jaatun}@uis.no

***Abstract*—Current proof-of-work blockchains are not sustainable in terms of energy needed to run them. In this paper we propose a new scheme that avoids wasted proof-of-work by a dynamic probabilistic method, where the consensus algorithm can be adjusted according to the parties' required assurance levels.**

## I. Background on Blockchain

The core ideas behind blockchain technology emerged in 1991 when a signed chain of information was used as an electronic ledger for digitally signing documents in a way that could easily show none of the signed documents in the collection had been changed [1]. In essence, a blockchain is a distributed ledger of cryptographically signed digital assets that are grouped into blocks, where each new block is cryptographically linked to the previous after validation and a consensus decision, such that modification of the assets in a block becomes more and more difficult as the blockchain extends [2]. The blockchain technology has enabled the success of many e-commerce systems such as Bitcoin, Ethereum, Ripple, and Litecoin. Because of this, blockchains are often viewed as bound to Bitcoin or possibly e-currency solutions in general. However, the technology is more broadly useful and is available for a variety of applications.

### A. The Blockchain Behind Bitcoin

As the core technology behind the cryptographic currency Bitcoin, blockchain was first conceptualized in a 2008 pseudonymous paper by Satoshi Nakamoto [3]. Bitcoin is an electronic payment system based on cryptographic proof, which allows any two willing parties to transact directly with each other without the need for a trusted third party.

In contrast with other e-payment schemes, Bitcoin was designed to be a completely peer-to-peer network, consisting of a large number of independent nodes which verify incoming transactions independently of each other. These nodes use a synchronization protocol that allows the nodes to agree on a common transaction history. Such a common transaction history is known as the *Bitcoin blockchain*, see Fig. 1 for a simplified version of a Bitcoin blockchain. By using the blockchain, Bitcoin became the first successful digital currency that solves the double-spending problem without the presence of a trusted intermediary.

As illustrated in Fig. 1, a Bitcoin block consists of two parts: block header and block data. A bunch of new transactions is collected into the data part of a block. All valid transactions in block data are hashed into a Merkle tree [4], [5]: copies of each transaction are hashed, which are paired and hashed iteratively until a single hash, known as the *Merkle root*, is ultimately generated (see Fig. 2). The Merkle root of all valid transactions is stored in the block header, which also stores the hash of the previous block's header, timestamps, nonce and other necessary information.

The blockchain of Bitcoin is maintained as follows:

1) the creation of a new block requires a large amount of work, known as *proof-of-work* (PoW);
2) peers are motivated by incentives (BTCs) in the network to create a new block ;
3) a new block is created after the difficult target for PoW is met;
4) the new block is appended to the existing chain of blocks through hash functions, which ensures immutability of the blockchain;
5) in case of branching, the longest chain that involves most PoW is accepted, and all other shorter ones are abandoned.

PoW is the critical part in the Bitcoin blockchain, which effectively prevents the issue of double-spending and the Sybil attack [6], in which a malicious entity could set up many peers that subvert the election and inject faulty information.

### B. Blockchain beyond Bitcoin

Bitcoin is by far the most widely known system that is intrinsically tied to blockchain technology. Although the Bitcoin itself is highly controversial, the blockchain technology has worked flawlessly and has a wide range of applications beyond Bitcoin [7].

As it evolves in new digital currencies and other applications, the original Bitcoin blockchain has been modified in different ways. However, after breaking down recent blockchain proposals, one can observe the following critical elements of blockchains:

- **a distributed network** - all blockchains are maintained in a distributed network without a central repository, where an identical copy of the ledger is shared across the network;

- **an immutable ledger** - the blocks in the ledger are cryptographically chained in chronological and linear

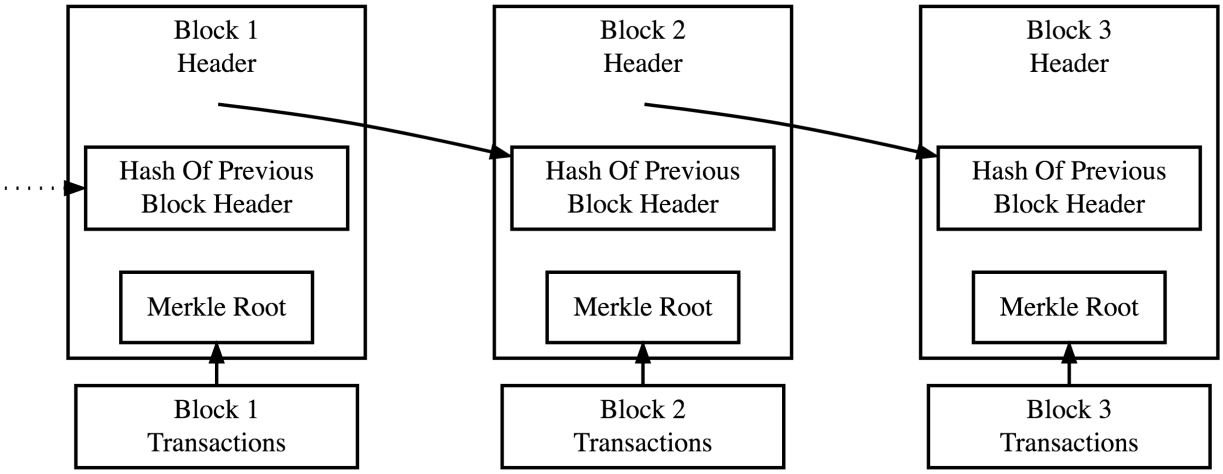

Fig. 1. Simplified Bitcoin Blockchain

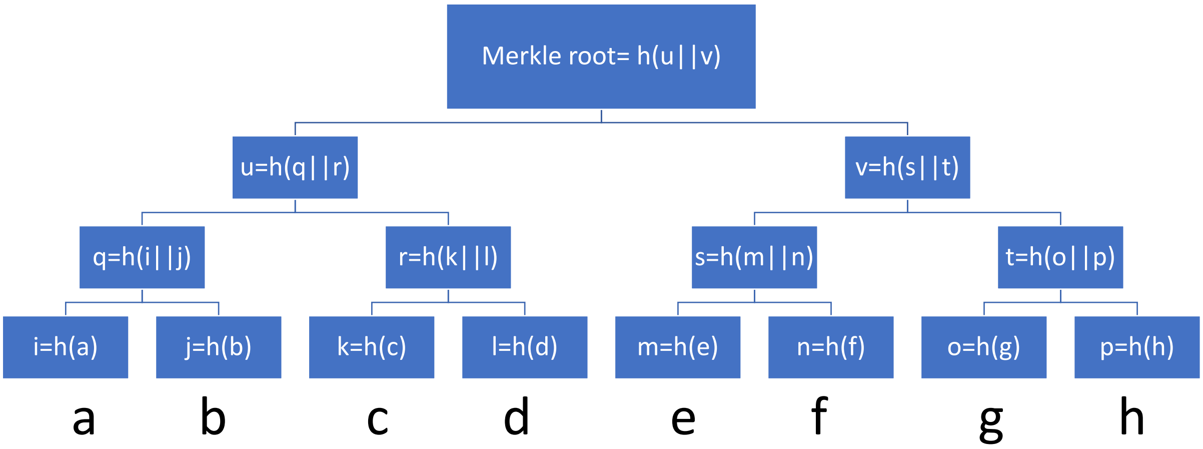

Fig. 2. Simple Merkle tree with Merkle root – a, b, etc. represent data chunks

order, which protects the digital transactions stored in the ledger from being modified;

- **a consensus protocol** - the shared ledger is maintained in a distributed fashion according to a consensus protocol. Consensus is a core concept in distributed systems. More concretely, a new block can be added to the blockchain only if it's validated by a pre-agreed consensus in the network. How to reach consensus in a distributed and trustless network is a variant of the Byzantine Generals (BGs) problem [8]. Proof-of-Work is the distributed consensus mechanism used in the Bitcoin blockchain. However, the PoW consumes a massive amount of CPU power to maintain the Bitcoin network. Therefore, other consensus mechanisms have been proposed in recent years to remedy the drawback of PoW, including (delegated) *proof of stake* (PoS) and *practical Byzantine fault tolerance* (PBFT), etc. Different consensus strategies are proposed from different considerations. The PoS consensus is based on the idea that the more stake a user has in the system, the more likely it will want the system to succeed [9], [10]. PoS-based blockchain systems use the amount of stake a user has as a determining factor for new block creation. The methods for how the blockchain system uses the stakes can vary – from random selection of staked users, to multi-round voting, to a coin aging system. Regardless of the exact approach, users with more stake are more likely to produce new blocks. The PBFT was proposed by Castro and Liskov [11] to tolerate Byzantine faults and made it practical to be used in even asynchronous environments, where the previous algorithms fell short. It is considered to be the first practical algorithm to achieve consensus when dealing with Byzantine faults. The PBFT was employed in the Hyperledger Fabric [12] as the primary consensus algorithm.

### C. Deficiencies of the PoW in Bitcoin blockchain

In Bitcoin, the pending transactions are stored in a transaction pool. For the incentive upon successful generation of a valid block, the mining nodes are competing on creating a hash less than a specified target, which is simply a pre-defined threshold. The effort required for generating a valid block is the already mentioned proof-of-work (PoW). However, as identified in the literature, this protocol suffers from several weaknesses:

**Waste of Resources.** Each PoW problem generally requires $10^8$ Giga hashes to be computed. The new ASIC machines used by the miners are built from scratch and are used to mine Bitcoins. These machines cannot serve any other purpose. Thus the total computing power spent on Bitcoin mining could theoretically be spent on other real world problems. More worryingly, the electricity consumed by Bitcoin mining machines has become a serious problem. A new study estimates that this process consumes at least 2.6 GW of power— almost as much electric power as Ireland consumes. Some speculate that this might rise to the level where it would account for almost half a percent of the world's electricity consumption [13].

**Unfair Competition.** A group of miners with large computational resources may set up a mining pool so as to control more than 50% of network computing power. In such a case the mining pool could launch a 51% attack, e.g., modifying the ordering or excluding the occurrence of transactions, or indulging a double spending by reversing transactions that they send. The mining pool keeps on earning maximum profit that leads to the socially undesirable problem of "rich get richer, poor gets poorer";

**Slow Block Creation.** The PoW protocol requires on average 10 minutes to generate a new block. Within the verification time a Bitcoin exchange might be completed. An attacker can send an illegitimate transaction log to the seller and another log to the Bitcoin network, where the buyer gets back his currency. By the time the seller realizes the fraudulent amount, the transaction may have been carried out. For this reason, the seller is suggested to wait for 6 new blocks are created before completing the exchange. This is unbearable in many circumstances, e.g., for in-person purchases;

**Selfish Mining.** Eyal and Sirer [14] showed that the network is vulnerable against selfish mining, where only a small portion of the computing power is used to cheat. The strategy of selfish mining is that selfish miners keep their mined blocks without broadcasting and release their private branches when some requirements are fulfilled. Before the publishing of the private blockchain, honest miners are wasting resources on a useless branch while selfish miners are mining the private branch without competitors. By doing so, the selfish miners tend to get more revenue once they are ahead of honest miners. It was shown by Sapirshtein et al. [15] that there are more advanced selfish mining strategies that are more profitable even for small miners.

## II. A NEW BLOCKCHAIN DESIGN

With the essential elements of the blockchain technology in mind, now we propose a new design of a public blockchain. The primary objective of our design is to address the resource-intensive issue of PoW in the Bitcoin-like blockchain protocols, where the distributed network is open to everyone and each node is equally treated. The new design shares a lot common elements and strategy with the known blockchains such as Bitcoin, and the core difference is the creation and validation of new blocks. Fig. 3 demonstrates the process of block creation in a nutshell.

### A. Terminologies

Before explaining the design in detail, we first introduce some terms that are commonly used in the literature. *Nodes* are connected to a peer-to-peer network and propagate new information via gossip[1] in the network. Each node keeps an identical copy of an order sequence of events in the form of a *blockchain* (The Bitcoin network defines *full nodes* and *light weight nodes*. Nodes that are responsible for creating new blocks in the network are termed *mining nodes* by convention. Here we define only full nodes for simplicity. But the light weight nodes can be defined in an exact same manner). *Accounts* in the network are identified by their public key or addresses obtained from the encoding of public keys. Each account can hold a sum of assets and can transfer its assets to any account in the network with *transactions*. The structure of each transaction is the same as that of Bitcoin. A transaction is valid only if all required rules are met, e.g., the corresponding account has valid balance (or inputting currencies) which are sufficient for the current transaction.

### B. Details of the design

The key idea of the design is that in each round of block creation, the mining node is dynamically assigned on basis of the input information of the previous block, the mining pool and the transactions in a random way.

As shown in Fig. 3, the process of block creation is composed of three phases:

**Phase 1.** This phase is the leftmost column of blocks in Fig. 3. At some point, when a valid block is added to the blockchain and a new block is to be created, the newly added block acts as a previous block, the pending transactions $\{TX_0, TX_1, \cdots, TX_{N-1}\}$ are added to a pool of transactions, which is implemented exactly the same as other Bitcoin-like cryptocurrencies.

However, there are two new features in the design. Firstly, an optional field, which is used to indicate the number of mining nodes that the sender and receiver of the transaction prefer to have in the mining phase, is added in each transaction. This field is left empty by default. If no number is specified, the sender and receiver will be believed to accept any number of mining nodes. If this field is filled with a number (agreed

[1] Gossip means that each node sends the information to all its neighbors, each neighbor in turn sends the information to all its neighbors, and so on.

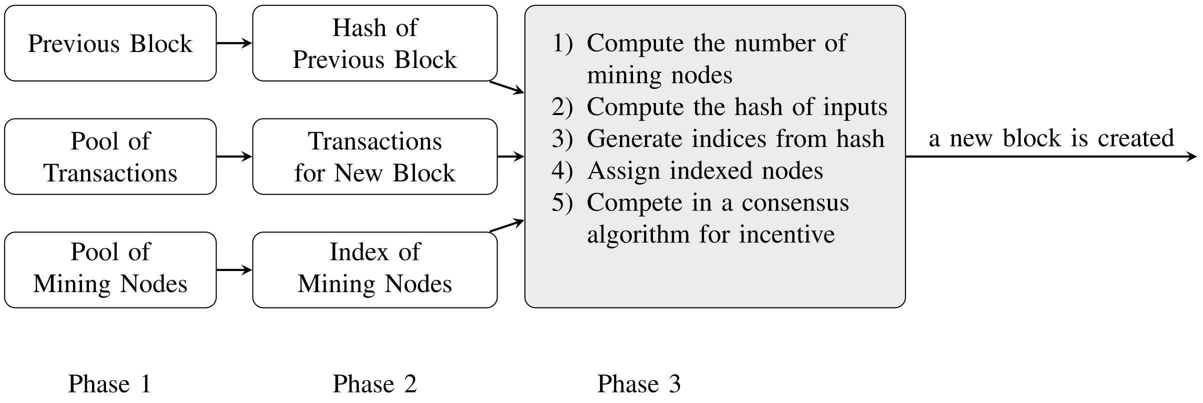
Previous Block
Hash of Previous Block
Pool of Transactions
Transactions for New Block
Pool of Mining Nodes
Index of Mining Nodes
1) Compute the number of mining nodes
2) Compute the hash of inputs
3) Generate indices from hash
4) Assign indexed nodes
5) Compete in a consensus algorithm for incentive
a new block is created
Phase 1
Phase 2
Phase 3

Fig. 3. Design of a cost-efficient block creation

by the sender and receiver), such a number will be an input to determine the final number of mining nodes in Phase 3. Secondly, we added a new component, termed as *mining registration* in the design, which is responsible for registering all nodes that plan to participate in the mining process into a pool of mining nodes. The implementation of this new component may be the same as the implementation of the pool of transactions in Bitcoin or other cryptocurrencies;

**Phase 2.** This phase is primarily for collecting input information for Phase 3. More concretely, the hash value of the previous block from a system-wide hash function, e.g., SHA256 or SHA-3, is computed; the system prioritizes the pending transactions according to a general strategy rule of priorities, e.g., the transactions with higher transaction fees and longer age in the pool are comparatively more prioritized and those with larger size are less prioritized, and collects a list of transactions $\{TX_{i_0}, TX_{i_1}, \cdots, TX_{i_{M-1}}\}$ with higher priorities and total size up to the limit of the data size in a block, and another list of to-be-added transactions $\{TX_{i_M}, TX_{i_{M+1}}, \cdots,\}$ for replacing invalid transactions in the first list; the system finally labels the nodes in the mining pool at the current stage as $Node_1, Node_2, \cdots, Node_n$;

**Phase 3.** This phase is the column in gray in Fig. 3. It is the crucial part of our design that assigns mining nodes in a sophisticated way:

1) the calculation of the number of nodes for the mining competition: the system collects all the transaction fee $Fee_j$ and the preferred number $Num_j$ in each transaction $TX_{i_j}$, $j = 0, 1, \cdots, M-1$ and compute the final number

$$k = \left\lceil \frac{\sum_{j=0}^{M-1} Fee_j * Num_j}{\sum_{j=0}^{M-1} Fee_j} \right\rceil \pmod{\delta n} + 1,$$

   where $\lceil x \rceil$ denotes the least integer equal to or greater than $x$ and $\delta$ is a threshold in the system and set to be 1 by default. If no prefer number is specified in all transactions $TX_{i_0}, \cdots, TX_{i_{M-1}}$ or $k$ happens to be zero (of which the probability is negligible), then the system randomly generated a number $k$ in the range of 1 to $n$.

   Here in the calculation of $k$ the transaction fee in each transaction is used as the weight of contributing its preferred to the determination of $k$. Such a consideration originates from many real-world scenarios - the voice of customers are heard more when they pay more.

2) the calculation of the hash: use the system-wide hash function, e.g., SHA256 or SHA-3, to calculate the hash value $H$ of the relevant inputs, including the hash of the previous block, the Merkle root of selected transactions, the current time-stamp, the identification/address of registered mining nodes;

3) the creation of index of mining nodes: based on the hash value $H$ in the previous step, the system first compute a binary string $s = s_0 s_1 \cdots = H||Hash(H)||Hash^2(H)|| \cdots$ until its length $l > k\tau$, where $\tau = \lfloor \log_2(n) \rfloor$; then the system derives $k$ pairwise integers from substrings $i_1 := s_0 \cdots s_{\tau-1}$, $i_2 := s_\tau \cdots s_{2\tau-1}$, $\cdots$, $i_k := s_{(k-1)\tau} \cdots s_{k\tau-1}$ (repeat the process if there are same integers among $i_1, i_2, \cdots, i_k$). The integers $i_1, i_2, \cdots, i_k$ are used as indices to select mining nodes in the next step;

4) the assigning of mining nodes: the system selects nodes $Node_{i_1}, \cdots, Node_{i_k}$ from the registered mining nodes $Node_1, \cdots, Node_n$ and assign them as mining nodes;

5) the consensus mechanism: if $k$ happens to be 1, then the lucky node $Node_{i_1}$ is directly responsible for the whole process of creating a new block, including verifying the validity of the transactions $TX_{i_0}, \cdots, TX_{i_{M-1}}$, replace invalid transactions with valid ones from the pool of pending transactions, create a block according to the pre-defined format of the blocks with an additional field including $k$, $n$ and $H$; if $k > 1$, the assigned $k$ mining nodes choose a consensus algorithm pre-defined in the system and compete for the incentives.
   Here the pre-defined consensus in the system is not neces-

sarily fixed. It could be any one of the known consensus algorithms, e.g., PoW [3], PoS [9] and PBFT [11]. Note that the number $k$ is highly likely to be smaller than the total number $n$ of nodes (or could be made to be much smaller than $n$ by setting the threshold). In this way, the process of randomly selecting $k$ mining nodes actually improves the known consensus algorithms. If the consensus algorithm is chosen to be PoW, then our design will be more fair and save a lot of computing power in the mining process, which will be discussed in detail later; if the consensus algorithm is chosen to be PoS, the issue of "The rich get richer and the poor get poorer" in PoS is also resolved to some extent since more randomness is added in the process.

After Phase 3, a new block will be created and ready to be appended to the blockchain after successful verifications by all other nodes. This process is the same as other blockchains.

## III. Analysis of the new blockchain

In this section we shall look into the design and further analyze it from different aspects.

**Flexibility.** As described in Phase 3 in Subsection II-B, the creation of a new block in our design is more like a framework than a specific design regarding the consensus algorithms. The set of consensus algorithms can be further extended when better rules are available. The system threshold for the number of mining nodes in each round can be also adjusted flexibly according to the throughput, network delay and performance of the whole system. In addition, the design currently focuses on the process of block creation, but doesn't concretely define the structure of a block and what kind of transactional data is to be stored in the blockchain. This flexibility allows for new structure of the block as well as a variety of transactional data;

**Security.** Security and efficiency are put on a pair of scales and balancing between them depends on the priority of applications. In our proposal, if $k$ is voted as one, a new block can be created at greatest efficiency and highest risk from security attacks; if $k$ is voted as $n$ and all mining nodes in the network have signed up in Phase 1, the creation of a new block is the same as Bitcoin network when PoW is chosen as the consensus algorithm. We now investigate the expected computing power controlled by an attacker in general. Denote by $\mathcal{P}$ the total computing power of the $n$ registered mining nodes. Suppose that the attacker controls $m$ out of $n$ ming nodes, of which the computing power $\mathcal{P}_{attack} = \delta \cdot \mathcal{P}$. From the generation of index set $I = \{i_1, i_2, \cdots, i_k\}$ for the assigned mining nodes, we can see that those $k$ mining nodes are picked out randomly. Hence, the expected computing power among $k$ nodes under the attacker's control is given by

$$E(P_{attack}) = \sum_{t=0}^{k} \binom{k}{t} \left(\frac{m}{n}\right)^t \left(\frac{n-m}{n}\right)^{k-t} E_t,$$

where $E_t$ is the expected computing power of random $t$ out of $m$ malicious nodes. For simplicity, we assume the malicious nodes have the same computing power at each node. In this case we have $E_t = \frac{t}{m} \cdot \mathcal{P}_{attack} = \frac{t}{m} \cdot \delta\mathcal{P}$, whence

$$\begin{aligned}
&E(P_{attack}) \\
=&\delta\mathcal{P} \sum_{t=0}^{k} \frac{t}{m} \binom{k}{t} \left(\frac{m}{n}\right)^t \left(\frac{n-m}{n}\right)^{k-t} \\
=&\frac{\delta\mathcal{P}}{m} \sum_{t=0}^{k} t \binom{k}{t} \left(\frac{m}{n}\right)^t \left(\frac{n-m}{n}\right)^{k-t} \\
=&\frac{\delta k\mathcal{P}}{n} \sum_{t=0}^{k} t \frac{(k-1)!}{(k-t)!t!} \left(\frac{m}{n}\right)^{t-1} \left(\frac{n-m}{n}\right)^{k-t} \\
=&\frac{\delta k\mathcal{P}}{n} \sum_{t=1}^{k} \frac{(k-1)!}{(k-t)!(t-1)!} \left(\frac{m}{n}\right)^{t-1} \left(\frac{n-m}{n}\right)^{(k-1)-(t-1)} \\
=&\frac{\delta k\mathcal{P}}{n} \sum_{t=0}^{k-1} \binom{k-1}{t} \left(\frac{m}{n}\right)^t \left(\frac{n-m}{n}\right)^{k-t} \\
=&\frac{\delta k\mathcal{P}}{n} \left(\frac{m}{n} + \frac{n-m}{n}\right)^{k-1} \\
=&\frac{\delta k\mathcal{P}}{n}.
\end{aligned}$$

Similarly, the expected computing power of random $k$ nodes is given by

$$\begin{aligned}
&E(P_{total}) \\
=&\sum_{t=0}^{k} \binom{k}{t} \left(\frac{m}{n}\right)^t \left(\frac{n-m}{n}\right)^{k-t} (E_t + \frac{(k-t)(1-\delta)\mathcal{P}}{n-m}) \\
=&E(P_{attack}) + \sum_{t=0}^{k} \binom{k}{t} \left(\frac{m}{n}\right)^t \left(\frac{n-m}{n}\right)^{k-t} \frac{(k-t)(1-\delta)\mathcal{P}}{n-m} \\
=&\frac{\delta k\mathcal{P}}{n} + \frac{(1-\delta)k\mathcal{P}}{n} = \frac{k\mathcal{P}}{n}.
\end{aligned}$$

From the above equations, we can see that the expected ratio of computing under the attacker's control is $\delta$. That is to say, if the computing power of malicious nodes and honest nodes are uniformly distributed, the expected ratio of computing power that the attacker controls for $k$ nodes is identical to that for $n$ nodes, which indicates that the expected ratio of computing power under the attacker's control is independent of the number of assigned nodes. For the cases where the computing power is not uniformly distributed, the expected ratio is difficult to estimate.

The above analysis provides a theoretic estimation on the threat of a 51% attack. In the Bitcoin network, the attacker with more than 50% computing power would always win the competition of proof-of-work in theory. In practice, implementing such an attack is definitely nontrivial. In our design, with the randomness of assigned mining nodes, the attacker can implement a successful 51% attack only if the malicious computing power of $t$ nodes is greater than the honest computing power of $k-t$ nodes, i.e.,

$$\frac{t}{m}\delta \cdot \mathcal{P} > \frac{k-t}{n-m}(1-\delta) \cdot \mathcal{P},$$

which is equivalent to

$$\frac{\delta}{1-\delta} \cdot \frac{t}{k-t} \cdot \frac{n-m}{m} > 1.$$

As opposed to the Bitcoin network where $\delta > 1 - \delta$ implies 100% success of attack, the above inequality implies that the success probability of the attacker depends on other factors, the number of malicious mining nodes in the network and the probability that $t$ malicious nodes are picked out from $k$ nodes, which is $\binom{m}{t}\binom{n-m}{k-t}/\binom{n}{k}$. From the attacker's perspective, the difficulty in implementing a successful 51% attack is increased by these factors.

**Performance.** As shown in Fig. 3, some new procedures were added in the process of creating a new block. In fact, all these procedures are very efficient and the overhead is completely eliminated by Step 5) of Phase 3. More concretely, in Phase 1 the only extra step is the sign-up of mining nodes in a pool, and in Phase 2 the only extra step is indexing mining nodes since the other two steps are done in all known proposals. In Phase 3, Step 1 results in some communication overhead in the network; Step 2 computes the hash on an input that is bound to all important information of the new block; the overhead of Steps 3-4 primarily comes from the generation of the index set $I$ for assigning mining nodes; Step 5 is the critical part of the design that tremendously contributes to efficiency as the number $k$ of mining nodes in Step 1 could be controlled as small as 1. In case of $k = 1$, the mining node only needs to verify the validity of selected transactions. In case of $k > 1$, the number $k$ is very likely much smaller than the total number of nodes in the network. Moreover, the consensus algorithm can be chosen to be rather efficient. For instance, the winning node is chosen to be the one which has the minimum Hash value in a pre-set time period instead of choosing the first one that obtains a hash value below the pre-set threshold.

**Cost-efficiency.** The strategy of randomly choosing a subset of mining nodes in the mining process significantly contributes to saving computing resource as well as electricity. As mentioned in the previous section, the electricity consumption of PoW in the Bitcoin blockchain and computation-intensive consensus algorithms in other blockchains has become a serious problem. This is one of the main motivations in this design. We aimed to reduce the waste of resources in the mining process while not significantly sacrifice the security. It's readily seen that our design indeed requires much less computations in the mining process, which contributes to a significant cost reduction.

To sum up, our new proposal is flexible with adjusting the number of mining nodes, the wining rules of mining and the structure of a block. When $k < n$ nodes are selected for the mining task, the efficiency is significantly improved while the expected security level is not sacrificed if the computing power of malicious nodes and honest nodes are uniformly distributed. In practice, the new design makes the attacker more difficult to control or predict the success probability of a 51% attack even though the attacker owns more than 50% of the computing power in the network.

## IV. Related Work

This section summarizes recent some known consensus algorithms that aimed to remedy the energy-wasting issue of PoW in the Bitcoin blockchain.

The PoS consensus algorithm was proposed primarily for handling the energy issue of PoW. This protocol trusts that if people have more currency involved (or at stake), they are less likely to attack the network. Miners with more stake will more likely be selected in the creation of new blocks. Miners in PoS need to prove the proprietorship of the amount of currency they possess. However, this selection method is unjust based on the research from the richest person in the network [9].

Delegated PoS (DPoS) is an algorithm like the PoS protocol. It varies from PoS algorithm in the aspect that in DPoS, coin holders of the cryptocurrency system vote for delegates to validate and process a transaction in return for transaction fees, which is different in PoS where a stakeholder validates and processes a transaction to earn rewards and transaction fees. DPoS leverages the power of stakeholder approval voting to resolve consensus issues in a fair and democratic way [10].

The PBFT algorithm does not scale, and it is mostly used in the permissioned or permission-based consortium blockchains. Recently, Luu et al. proposed a novel design of permissionless blockchain with the Bazantine-like consensus [16]. The main idea of the design is to divide a network into small chunks known as committees. Each committee processes a disjoint set of transactions and the whole procedure is parallelized and the agreement condition is implemented in probabilistic manner. Each honest process matches its agreed value with a constraint function and checks its validity, and the solution is accepted only if it satisfies the constraint function. All committees perform a classic Byzantine consensus and the final committee merges the results of all previous committees.

## V. Conclusion

In this paper we propose a new blockchain scheme for Bitcoin-like cryptocurrencies which require a public and trustless distributed network. Several new components and procedures were added in the blockchain to reduce the number of mining nodes for creating new blocks and randomize the selection process of such nodes. We have shown that the proposed scheme gains advantages in flexibility, efficiency and resource-consumption over Bitcoin-like blockchains while not significantly sacrificing security. The proposed scheme can be further extended to other applications of blockchain with smart contracts.

## Acknowledgement

This work was supported by the research project (No. 720025) from UH-nett Vest in Norway.

# References

[1] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[2] M. Risius and K. Spohrer, "A blockchain research framework," *Business & Information Systems Engineering*, vol. 59, no. 6, pp. 385–409, Dec 2017.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf," 2008.

[4] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[5] D. Bayer, S. Haber, and W. S. Stornetta, *Improving the efficiency and reliability of digital time-stamping*. New York, NY: Springer New York, 1993, pp. 329–334.

[6] J. R. Douceur, "The sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 251–260.

[7] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, 2017.

[8] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4/3, pp. 382–401, July 1982.

[9] D. Larimer. Transactions as proof-of-stake. [Online]. Available: https://bravenewcoin.com/assets/Uploads/TransactionsAsProofOfStake10.pdf

[10] D. Hallberg, A. Balzini, D. L. Borella, and D. Calderoni. White paper - proof of stake. [Online]. Available: https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[11] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.

[12] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, 2016.

[13] A. de Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801 – 805, 2018.

[14] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, Jun. 2018.

[15] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *Financial Cryptography and Data Security*, J. Grossklags and B. Preneel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 515–532.

[16] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 17–30.