EPoW: Solving Blockchain Problems Economically

Chih-Wen Hsueh
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan, R.O.C.
Email: cwhsueh@csie.ntu.edu.tw

Chi-Ting Chin
Department of Risk Management and Insurance
Ming Chuan University
Taipei 111, Taiwan, R.O.C.
Email: debbyjin@zeta.mcu.edu.tw

***Abstract*—Blockchain was first implemented in Bitcoin, the first decentralized digital currency, with the hash-chained blocks of data, hash-based proof-of-work, and a peer-to-peer protocol to reach consensus so as to append new blocks and secure transactions. Most importantly, it prevents double-spending after some confirmation time. However, few countries accept Bitcoin as a legal currency because, in addition to being subject to some security attacks, there are still problems in economics, politics, performance, etc. Many altcoins with different improvements have been proposed, but problems and attacks still remain. Meanwhile, blockchain, as a trust machine, has been applied to many other areas, such as smart contract, creating a new hype, called "the most significant innovation since the Internet." However, there are still similar problems and attacks in those applications. By changing the original consensus design, instead of just providing proof of work, we propose *EPoW*, i.e. estimable proof-of-work, to estimate how much work is done and a corresponding protocol to reach consensus. *EPoW* and the protocol can serve as a new instrument such that the problems and attacks mentioned above might be relaxed or solved economically. With *EPoW*, we believe blockchains and digital currencies can be better customized by companies, organizations or countries for different purposes or policies and be employed by any users.**

I. Introduction

Blockchain was named after the technology was first proposed in the Bitcoin white paper[11]. It was implemented later in Bitcoin, the world's first decentralized digital currency[1][20], and the source code is open[21]. Blockchain[22] consists of a distributed database that maintains continuously growing records of data called blocks. Each block contains a timestamp of its birth, a link to a previous block and a hash[33] value of the whole previous block. A hash function in cryptography is a one-way function to map data of arbitrary size to data of fixed size such that it is very difficult to reconstruct the input data from the output hash value. Therefore, these blocks are hash-chained, forming the so-called blockchain, which is inherently resistant to modification of the data. In other words, any modification in a block results in inconsistency of its hash value in the next block, as well as in the following blocks.

In Bitcoin, users can run the Bitcoin program at any computer nodes on the Internet[6], forming the Bitcoin network. The program has many options to choose for applying Bitcoin rules. Since the source code is open, users can also customize their own code as long as it conforms to the Bitcoin rules. A full node in Bitcoin network downloads every block and transaction and checks them against core consensus rules. A full-node user, called a miner, follows all Bitcoin rules, while other users depend on miners to only send and receive transactions, taking less responsibility and workload. At first, miners get a copy of all blocks of the blockchain from full nodes. Users send transactions to each other by broadcasting the transactions as Internet packets to miners for confirmation. Miners simultaneously collect broadcast transactions and each one works on his/her own new block. By completing a hash-based proof of work (PoW), any miner can get the right to append a new block to the blockchain. By a consensus protocol, the miners will decide which new blocks will be accepted in the blockchain. The process is called mining because the miners get rewards once the block is confirmed. If a block is confirmed, fixed amount of bitcoins, decreasing by half every 4 years, and total transaction fees of the transactions recorded in the block will be given to the miner as a block reward.

By trying a number, *nonce*, in its own mining block, the PoW asks the miners to find a hash value of the mining block to be less than a number, *target*. Once the nonce is found, the new block is mined and waits for confirmation. The smaller the target value, the more difficult to mine a block. The target value is adjusted every 2016 blocks mined so that the time to mine a block, called the block interval, has an average of 10 minutes. A number proportional to the target value, called *difficulty*, represents the total mining computing power. Another number, *hash rate*, the estimated total hash rate of the blockchain, also indicates the total mining computing power. However, no indicator represents the mining computer power of individual miners unless we do long-term statistical analysis of the blocks mined. As miners race to reach consensus earlier, the difficulty value grows exponentially, and so does the power consumption. Since the beginning of Bitcoin, the year of 2009, until June 2017, the difficulty of Bitcoin grew from 1 to over 5.95×10^{11}[25]. Suppose only one computer was used in the beginning of Bitcoin network. Nowadays roughly every person in the world simultaneously uses more than 79 of the same computers for mining. More charts of Bitcoin statistics are also available online[26].

The mined block records the transactions the miner chose before mining, e.g. the earlier ones or the ones with higher transaction fee, and the block might be broadcast to the other miners for confirmation. Note that the mined block might be later or never broadcast. If the block and its transactions

have been verified by a miner, the miner might accept and confirm the block to the blockchain, then continue to mine for the next block. The transactions in the confirmed block will be removed from the transaction pool of the verifying miner to mine for the next block. Note that each miner might reject the arriving new blocks and continue mining his/her own block. Confirming a block means appending it to the local copy of blockchain, allowing for retrieval by other users. A transaction has one confirmation when its block is confirmed. For each subsequent block confirmed, the transaction has one more confirmation. Because of the concurrency, block mining protocol and inevitable network delay, the miners might accept more than one blocks before mining the next one. Actually, blocks with a timestamp greater than 2 hours from the current network time and earlier than the median time of the past 11 blocks are rejected. This can reduce the number of confirmation to avoid attacks[19].

The very first block is called the genesis block. The main chain of a blockchain is the chain of blocks with the largest accumulative difficulty value from the genesis block. The honest miners always choose to append a new block at the end of the main chain. Therefore, the miners might have different views of a blockchain and a blockchain might fork a branch sometimes. However, after some block confirmations, the main chain will be consistent. The forked blocks not in the main chain will be still kept in the blockchain. Since only one main chain is maintained and all transactions are verified by peer users, double-spending [8][17] of a transaction can be prevented after some confirmations. Therefore, Bitcoin blockchain is an open decentralized ledger that can securely record transactions between any two parties. Of course, the less confirmation time, the higher the satisfaction of users.

While only few countries accept bitcoin as a legal currency, most others have banned or restricted it, e.g. as a kind of expensive goods. Because, in addition to some unsolved attacks, there are still problems in economics, politics, performance, etc., more briefed in Section II. To win the mining race, some miners adopt hardware assistance, such as GPU or ASIC, aggravating the growth of difficulty. The ASIC, application-specific IC, runs even faster than GPU, graphics processing unit. Without constraining the growth of difficulty, the computing power consumed for mining keeps growing exponentially. Recently, the Bitcoin network consumes over two million US dollars daily in electricity[29]. By 2020, it could consume as much electricity as Denmark[27]. However, most of the time, there are only fewer than 10 transactions per second (TPS)[38]. The computing is very not green at all.

In Ethereum[31], a new PoW, Ethrash[30], which is ASIC resistant but GPU favorable, was proposed to reduce mining racing. A new consensus protocol, GHOST[42], was also proposed to shorten block interval into 12 seconds. The confirmation time is reduced but the TPS is still low, even fewer than 1. Other altcoins also propose different consensus protocols, such as proof of stake[37], where altcoins are Bitcoin alternatives with uninteresting small changes. However, departing from the original definition of proof of work, they might not reflect that there is enough work done to fulfill the original request, more briefed in Section II. Instead of just serving for digital currencies, blockchain was recently applied to many other areas, such as smart contract[41] in Ethereum. A smart contract is a computerized transaction protocol that executes the terms of a contract. Ethereum blockchain enforces the correct execution of smart contracts in blockchain transactions on Ethereum Virtual Machine. Blockchain already creates a new hype, called "the most significant innovation since the Internet"[28]. However, similar problems and attacks are left to be solved because some influential factors such as how much work has been done by individuals are not known or are inestimable in real-time.

To reach consensus, Bitcoin adopts proof of work and how much work is done individually can only be estimated in average in a long run. Our main contribution is as follows:

- We first propose and prove *EPoW*, i.e. estimable PoW, where it can quantitatively estimate how much work is done individually in closed formulas in real-time.
- We propose a consensus protocol to apply *EPoW*.
- We conduct simulations to measure the variance and standard deviation of estimations.

The new PoW and corresponding consensus protocol might serve as a new instrument such that the blockchain problems and attacks might be relaxed or solved economically. We have not found similar related work.

The rest of this paper is organized as follows. Section II provides a background brief. The *EPoW* is introduced in Section III. Section IV describes how to use it. Section V provides some discussion. The paper is concluded in Section VI. Most on-line references are ended with the last accessed date. The undated are last accessed as of June 10, 2017.

II. Background

This is only a brief. Detail surveys are available[12][13].

A. Proof of Work

PoW is represented by a piece of data sent from a requester to the PoW service provider. The key feature of PoW is the work must be moderately hard (but feasible) on the requester side but easy to check for the service provider[36]. It was first proposed to prevent junk mails[2] by doing some significant work before sending the email. For the hash-based PoW as in Bitcoin, the PoW is to prove that enough computing work has been done so that the winner can append a new block. The same principle requiring certain amount of "work" applies to proof of space, proof of bandwidth, and proof of ownership as well. However, proof of stake is different in that it just needs to provide a more deterministic proof of "wealth", e.g. "age" of coin or how long a coin has been created.

B. Consensus protocol

With the proof of work, proof of stake, or other means, a consensus protocol is to decide who or what wins the right to append a block, write a ledger, or do something else. For example, in Bitcoin, the first miner with a low hash value

wins and appends the block on the main chain. Byzantine fault tolerance (BFT) is another consensus protocol by voting in a faulty distributed system[23]. It might not be very scalable because there are a lot of voting messages, which might be faked, and it needs to wait for most of the messages to arrive to make a decision. Moreover, BFT miners need to be identified, i.e. they are permissioned.

C. Types of Blockchains

1) public blockchain: Anyone can participate.
2) consortium blockchain: The miner is permissioned[9].
3) private blockchain: Not anyone can participate.

To be really trustful and decentralized without colluding, any user should be able to participate freely. However, to be trackable or responsible, user identification might be necessary but this is against privacy. Therefore, semi-permissioned where the identification can be checked in a decentralized and encrypted manner is desirable.

D. Weaknesses

For those systems using randomization to reach fairness in the long run, attacks might happen if we do not wait enough time for confirmation. For example, hashing of PoW in Bitcoin is similar to random number generating[3]. The PoW has at least the following weaknesses[43].

1) 51% attack: One party has more than 50% total mining power. Eventually, it can redefine the main chain, where data are still immutable but some could be excluded.
2) double-spending: The same transaction is issued again after the first transaction has finished, e.g. cashout, before enough confirmations. It is done by redefining the main chain to exclude the first transaction.
3) Sybil attack: Divide oneself into more participants. It might forge identification.
4) Selfish mining: Delay broadcasting of mined block to get more percentage of average reward than honest ones[14].
5) Eclipse attack: Some nodes might be isolated by neighbors delaying new arriving blocks[7][16].
6) Balance attack: In a consortium blockchain using BFT, double-spending might happen with much lower than 50% of mining computing power in two groups of nodes with about the same computing power[5][12].

E. Problems

1) politics: The government would not like to be exposed to attacks, let anyone to participate in operating, or lose control of a legal currency.
2) economics: Terms such as rewards, fees, or taxes lack quantifiers of user work, government policies, etc. to form a complete economic formula or model.
3) performance: Hundreds or thousands of times of TPS might be needed[39]. Blockchain size is too big and, for general use, it can not keep growing unlimitedly[15][18].
4) privacy: Any user can see any data in blockchain but most users would not like to reveal private information.
5) crime: Eventually, faulty system design or programming bugs might be fixed or controlled to reduce illegal behavior. However, for some behavior, there might be a legal loophole. For example, coin mixing[35] can improve privacy but it is legal only if the coin source is legal.

F. Challenges

The challenges mostly come from dishonest adversaries. For blockchains to be a real trust machine, the permissioned approach is questionable because anyone might be somewhat dishonest sometimes, especially when the identification is known to act in collusion. However, if we allow anyone without identification to participate, we need to know them more quickly before they do any damage. Data mining on a blockchain can help a lot to retrieve information to improve the security, but it might be too late. How to know participants better in real-time with little information is challenging. One metric might be how much they had worked in the PoW system. This is also the motivation of *EPoW*.

Zero-knowledge proof[44] was applied in blockchain to enforce honest behavior while maintaining privacy. It is also challenging because it needs a lot of computing power.

III. Estimable Proof-of-work

EPoW is an estimable PoW such that how much work was done can be estimated in closed-form formulas. The requirement of a closed-form formula arises from the concern that the numbers involved are usually very big. According to the source code, Bitcoin uses the hash function SHA256[40], with hash range $2^{256} \cong 1.158 \times 10^{77}$. Since the work is hard by definition, it is not clear whether there always exists a closed-form estimation.

For the PoW in Bitcoin, the service provider can easily check that some hard work was done by the service requester but does not know how much work has been done quantitatively. If we would like to convert the PoW into *EPoW*, instead of just providing one nonce value as PoW, two nonce values, i.e. *low nonce* and *high nonce* are needed in *EPoW*. The low nonce generates a hash value, called *low hash*, less than the other hash value, called *high hash*, generated by the high nonce in the same block mining. The value range from low hash to high hash is called *trial range*. If low hash and high hash are the lowest hash value and the highest hash value ever generated, respectively, we can estimate how many times the nonces have been tried, called *nonce trials*, by trial range. It is an estimation of how much work has been done and the estimation can be figured out in closed-form formulas.

If the low hash is not the lowest one, the high hash is the not highest one, or the same nonce is tried more than once, the nonce trials might be underestimated. If we grant the miner of larger nonce trials more rewards, honest miners would not like to underestimate the nonce trials. Note that underestimating on purpose might be a new attack. Long-term statistics might be able to relax the underestimation. Since we need to try nonces to get the desired hash values, it is very difficult to

fake untried lower or higher hash values. Because nonce trials increase with trial range, it is very unlikely to overestimate nonce trials. Before we prove the correctness of *EPoW*, we need to prove the following Lemma.

Lemma III.1. *Suppose integers* 1 *to* N *are uniformly generated with the same probability* $p = 1/N$*. At the* m*-th generation,* $P(i, j|m)$*, the probability of the minimum ever generated being* i *and the maximum being* j*, where* $1 \leq i \leq j \leq N$ *and the range* $n = j - i + 1$*, is*

$$\begin{cases} p^m, & i = j \\ (n^m - 2(n-1)^m + (n-2)^m)p^m, & \textit{otherwise} \end{cases}.$$

Proof. It is a Markov process with a lot of states but can be solved easier by the following analysis of difference equations. For the first generation, only one number is generated. It is trivial $i = j$ and $P(i, j|1) = p$. These are the initial values of the difference equations. For the following generations, any number is generated with a probability p. Starting from the second generation, if $n = 1$, it is trivial only the same number is generated and $P(i, j|m) = p(P(i, j|m-1))$, otherwise

$$P(i,j|m) = p(nP(i,j|m-1) + \sum_{k=i}^{j-1} P(i,k|m-1) + \sum_{k=i+1}^{j} P(k,j|m-1)). \quad (1)$$

To the right of equal sign in Equation (1), there is a single term (the first part) followed by two summations (the second part). The first part accounts for the cases where the next generation retains the same range, while the second part covers the cases where either the maximum or the minimum is breached. Let ii and ij stands for the row index and column index of a matrix, respectively. The equations can be expressed as the following matrices, where D_N is the Markov transition matrix: $P_m =$

$$[P(i,j|m)] = \begin{bmatrix} P(1,1|m) \\ P(1,2|m) \\ \vdots \\ P(1,N|m) \\ P(2,2|m) \\ \vdots \\ P(2,N|m) \\ \vdots \\ P(N,N|m) \end{bmatrix} = pD_N \begin{bmatrix} P(1,1|m-1) \\ P(1,2|m-1) \\ \vdots \\ P(1,N|m-1) \\ P(2,2|m-1) \\ \vdots \\ P(2,N|m-1) \\ \vdots \\ P(N,N|m-1) \end{bmatrix}.$$

Let $L_0 = []$, and

$$L_k = \begin{bmatrix} \lambda_k & 0_{\frac{k(k-1)}{2} \times \frac{k(k-1)}{2}} \\ 0_{\frac{k(k-1)}{2} \times \frac{k(k-1)}{2}} & L_{k-1} \end{bmatrix}_{\frac{k(k+1)}{2} \times \frac{k(k+1)}{2}},$$

where $\lambda_k = [\lambda_{ii,ij}] = \begin{bmatrix} \begin{cases} ii, & ii = ij \\ 0, & \text{otherwise} \end{cases} \end{bmatrix}_{k \times k}$, to define λ or L_N, the matrix of eigenvalues of D_N. For example,

$$L_4 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0 \\ 0&2&0&0&0&0&0&0&0&0 \\ 0&0&3&0&0&0&0&0&0&0 \\ 0&0&0&4&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&2&0&0&0&0 \\ 0&0&0&0&0&0&3&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&2&0 \\ 0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \text{ and}$$

$$D_4 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0 \\ 1&2&0&0&1&0&0&0&0&0 \\ 1&1&3&0&0&1&0&1&0&0 \\ 1&1&1&4&0&0&1&0&1&1 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&2&0&1&0&0 \\ 0&0&0&0&1&1&3&0&1&1 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&2&1 \\ 0&0&0&0&0&0&0&0&0&1 \end{bmatrix}.$$

Note that the matrix dimension is very big when $N = 2^{256}$. Precisely, let $D_0 = []$, and

$$D_k = \begin{bmatrix} d_k & R_{k,k-1} \\ 0_{\frac{k(k-1)}{2} \times k} & D_{k-1} \end{bmatrix}_{\frac{k(k+1)}{2} \times \frac{k(k+1)}{2}}, \text{ where}$$

$$d_k = \begin{bmatrix} \begin{cases} \lambda_{ii,ij}, & ii = ij \\ 1, & ii > ij, 2 \leq ii \leq k \\ 0, & \text{otherwise} \end{cases} \end{bmatrix}_{k \times k} \text{ and}$$

$$R_{k,l} = \begin{bmatrix} \begin{bmatrix} \begin{cases} 1, & k-l+1 \leq ii = ij+1 \leq k \\ 0, & \text{otherwise} \end{cases} \end{bmatrix}_{k \times l} & R_{k,l-1} \end{bmatrix}.$$

Since D_N is invertible, it can be eigendecomposed as $X\lambda X^{-1}$, where X is the eigenmatrix of D_N. $X = X'_N = [X_{i,j}] = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,N} & X_{2,2} & \cdots & X_{2,N} & \cdots & X_{N,N} \end{bmatrix}$. $X_{i,j}$ is a column vector with size $\frac{N(N+1)}{2}$. For example,

$$X'_4 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0 \\ -1&1&0&0&-1&0&0&0&0&0 \\ 0&-1&1&0&1&-1&0&0&0&0 \\ 0&0&-1&1&0&1&-1&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&-1&1&0&-1&0&0 \\ 0&0&0&0&0&-1&1&1&-1&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&-1&1&-1 \\ 0&0&0&0&0&0&0&0&0&1 \end{bmatrix},$$

$$X_{1,1} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad X_{1,2} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \cdots. \text{ Precisely, let}$$

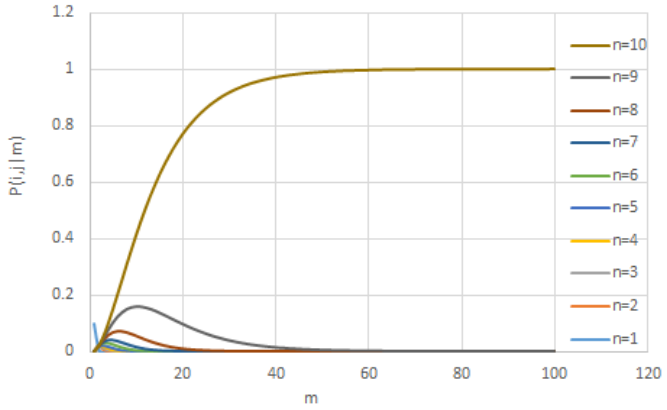

Fig. 1. The probability of less than 100 nonce trials with $N = 10$.

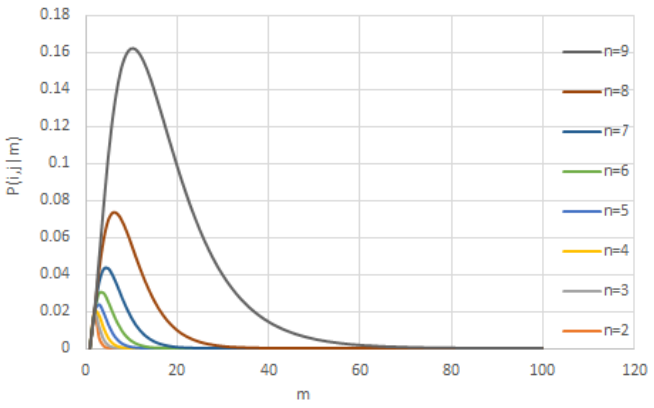

Fig. 2. Normal cases of Figure 1.

$X'_0 = [],\ X'_k = \begin{bmatrix} x_k & X''_k \\ 0_{\frac{k(k-1)}{2}\times k} & X'_{k-1} \end{bmatrix}_{\frac{k(k+1)}{2}\times\frac{k(k+1)}{2}}$, where

$$x_k = \begin{bmatrix} 1, & ii = ij \\ -1, & 2 \le ii = ij+1 \le k \\ 0, & \text{otherwise} \end{bmatrix}_{k\times k} \text{ and}$$

$$X''_k = \begin{bmatrix} -1, & 2 \le ii = ij+1 \le k \\ 1, & 3 \le ii = ij+1 \le k \\ 0, & \text{otherwise} \end{bmatrix}_{k\times\frac{k(k-1)}{2}}.$$

The initial conditions are $P_1 = [P(i,j|1)] = \begin{bmatrix} p, & i=j \\ 0, & \text{otherwise} \end{bmatrix}$. The difference equations can be simplified as follows $P_m = pD_N P_{m-1} = p^{m-1}X\lambda^{m-1}X^{-1}P_1 = p^m\sum_{i=1}^N\sum_{j=i}^N c_{i,j}\lambda_{i,j}^{m-1}X_{i,j}$. Solving for $m = 1$, $P_1 = p\sum_{i=1}^N\sum_{j=i}^N c_{i,j}X_{i,j}$, we derive $c_{i,j} = \lambda_{i,j}$.

Therefore, $[P(i,j|m)] = p^m\sum_{i=1}^N\sum_{j=i}^N c_{i,j}\lambda_{i,j}^{m-1}X_{i,j} = \begin{bmatrix} p^m(1^m), & i=j \\ p^m(-2(n-1)^m + n^m + (n-2)^m), & \text{otherwise} \end{bmatrix} = \begin{bmatrix} p^m, & i=j \\ (n^m - 2(n-1)^m + (n-2)^m)p^m, & \text{otherwise} \end{bmatrix}$.

$P(i,j|m)$ is only dependent on n and m, while p is a constant, $1/N$. Note that we add one more item $(n-2)^m$ for $n = 2$ to make the final formula with only 2 forms for easier application later. We can check that $\sum_{i,j} P(i,j|m) = 1$. □

Assume the hash function used in PoW is perfect, and it behaves like a fair random number generator. According to Lemma III.1, using *EPoW*, the probability of the trial range after m-th trial has a closed-form formula. As shown in Figure 1, w.l.o.g. for easier description with $N = 10$ and $m < 100$, the general shape will be similar regardless of N and with big enough m.

Except the biggest range $n = N = 10$, the other probabilities are relatively very low and soon drop near 0 after about 50 trials. However, as N increases, if m is relatively small, the probability of $n = N$ are also very low. The current, June 2017, hash rate of Bitcoin network, 3.4×10^{21} hashes per 10 minutes, can be a good reference value of m. It is relatively small compared to $N \cong 1.158 \times 10^{77}$, and we shall focus on the very left part of Figures 1 and 2.

TABLE I
THE STATISTICS OF $P(i,j|m)$.

$n = j - i + 1$	$n = 1$	$2 < n \le N$
$P(i,j\|m)$	p^m	$m = 1$: 0; $m > 1$: $(n^m - 2(n-1)^m + (n-2)^m)p^m$
$P(n\|m)$	p^{m-1}	$(N-n+1)(n^m - 2(n-1)^m + (n-2)^m)p^m$
$\bar{n} = \sum_{n=1}^N nP(n\|m)$	$N - 2p^m S_m(N-1),\ \int_0^n x^m < S_m(x) = \sum_{k=1}^x k^m < \int_1^{n+1} x^m$	
$P(m\|i,j)$	$P(i,j\|m)/P(i,j)$	
$P(i,j) = \sum_{m=1}^M P(i,j\|m)$	$f_1(p)$	$f_1(np) - 2f_1((n-1)p) + f_1((n-2)p)$
	$f_1(x) = \sum_{k=1}^M x^k = \begin{cases} M & , x = 1 \\ (x - x^{M+1})/(1-x), & \text{otherwise} \end{cases}$	
$\bar{m} = \sum_{m=1}^M mP(m\|i,j)$	$f_2(p)/P(i,j)$	$(f_2(np) - 2f_2((n-1)p) + f_2((n-2)p))/P(i,j)$
	$f_2(x) = \sum_{k=1}^M kx^k = \begin{cases} \frac{M(M+1)}{2} & , x = 1 \\ (f_1(x) - Mx^{M+1})/(1-x), & \text{otherwise} \end{cases}$	
$Var(\bar{m}) = \sum_{m=1}^M m^2P(m\|i,j) - \bar{m}^2$	$f_3(p)/P(i,j) - \bar{m}^2$	$(f_3(np) - 2f_3((n-1)p) + f_3((n-2)p))/P(i,j) - \bar{m}^2$
	$f_3(x) = \sum_{k=1}^M k^2x^k = \begin{cases} \frac{M(M+1)(2M+1))}{6} & , x = 1 \\ (2f_2(x) - f_1(x) - M^2x^{M+1})/(1-x), & \text{otherwise} \end{cases}$	

For clarity, Figure 2 is Figure 1 scaled after removing the rare $n = 1$ and $n = N$ cases. For each n, the m with the highest probability could be a quick estimation of nonce trials. Since the probability distribution skews a lot, other estimation might be more convincing to distinguish how much work has been done. Actually, $\bar{m}$, the average nonce trials might be a better estimation.

As shown in Table I, we have different statistics by $P(i,j|m)$. $P(n|m)$ is the probability of a given trial range n after the m-th trial. Since $\sum_n P(n|m) = 1$, the m value, where the average n, $\bar{n} = \sum_{k=1}^N kP(k|m)$, is equal to the given trial range, might be a good estimation of $\bar{m}$. Unfortunately, $\bar{n}$ contains a Faulhaber's formula[32], $\sum_{k=1}^N k^m$. When N is large, it is not computationally feasible. However, approximating in closed-form formulas is still possible, as the

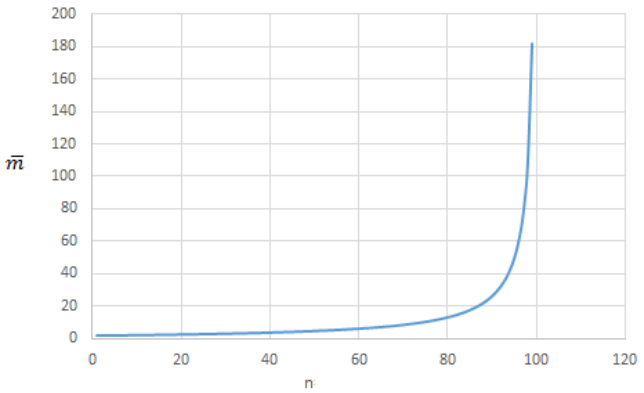

Fig. 3. The work estimation $\bar{m}$ with $N = 100$.

integral bounds of $S_m(x)$ shown in Table I, providing that some computation is acceptable.

Assuming the trials are done uniformly at a fixed rate as in Lemma III.1, the probability of m-th trial, $P(m)$, is a constant. By Bayes' theorem, the probability of nonce trials on given i and j, $P(m|i,j) \cong P(i,j|m)P(m)/P'(i,j) = P(i,j|m)/P(i,j)$, where $P'(i,j) = \sum_{m=1}^{\infty} P(i,j|m)P(m)$ and $P(i,j) = \sum_{m=1}^{\infty} P(i,j|m)$. $\bar{m} = \sum_{m=1}^{\infty} mP(m|i,j)$ is a better estimation and a closed formula. As shown in Figure 3, the estimation of trials $\bar{m}$ grows exponentially, as the trial range n goes near 100, when $N = 100$. In practice, N is very big, so is n. Therefore, there should be enough values of $\bar{m}$ to distinguish how much work was done.

Since the trials are basically random in nature, the variance is large. In our simulation, $N = 2^{256}$, the standard deviation of $\bar{m}$ grows linearly from 0 to 0.63 times of $\bar{m}$, when n goes to N. The 95% confidence interval is around 1 to 4 standard deviations. The variance can be lowered if we allow removing the extreme, too big or too small, samples. Note that we need to extend the GMP big number library in 64bit Linux to calculate the big numbers because the exponent is only an unsigned long integer, which is not big enough.

Theorem III.2. *Hash-based* EPoW *is feasible and estimable.*

Proof. For hash-based PoW, it follows Lemma III.1 and the probability for estimation is in closed-form formulas. Assuming the hash-based trials in proof of work are conducted at a constant rate, so that the Bayes' theorem can be applied to estimate the average trials. We can map the estimation values into different groups and assign a new representing value for each group, so that the number of representing values are limited. In the worst case, a table of the representing values with an acceptable look-up complexity can be built. Therefore, it is feasible and estimable. □

Since we only would like to estimate how much work was done, it does not need to be very accurate. Actually, there might not exist exact numbers to represent how much work was done. However, an instrument to distinguish quantitatively is definitely necessary. For non-hash-based PoW, if the estimation can be in closed-form formulas, *EPoW* is also feasible.

IV. Application to *EPoW*

EPoW provides an estimation value indicating how much work has been done. The value might represent the mining reward directly but in the real world many factors are not reflected. For example, the Cobb-Douglas production function[24](CDPF), suggests $Y = AL^{\beta}K^{\alpha}$, where

- Y = total production (the real value of all goods produced in a year)
- L = labor input (the total number of person-hours worked in a year)
- K = capital input (the real value of all machinery, equipment, and buildings)
- A = total factor productivity

  α and β are the output elasticities of capital and labor, respectively. These values are constants determined by available technology.

Even though the CDPF is a simplification of the real world, it is widely used. We further simplify it here for easier discussion. Suppose AK^{α} can be a constant 1 relative to the labor input L^{β} in a period of time, i.e. $Y = L^{\beta}$, and the production in PoW can be linearly rewarded (or sold). The work can be linearly quantified as labor input and the work in Bitcoin blockchain can also be directly measured by computing power. Then, the reward and computing power in Bitcoin blockchain should also follow CDPF. Since Bitcoin provides the miners a constant block subsidy and variable transaction fees as reward, it is reasonable that we only control the percentage of the reward granted to the miners. Therefore, we assume the mining reward is 1 or 100% initially, scaling it by a better distribution of the reward afterward. L can thus stand for the estimated nonce trials $\bar{m}$ and Y stands for the control percentage.

Because of the hash-based randomization, the probability to get the reward is linear to computing power in Bitcoin. Therefore, Bitcoin has $\beta = 1$. It is an ideal number that miners would not like dividing their computing power similar to the Sybil attack, nor combining into a mining pool as a monopoly because the average reward is the same in the long run, no matter how the difficulty grows. However, in reality, miners join mining pools because the facility might be more cost effective than individuals and miners can save the maintenance cost.

In the law of diminishing marginal returns, the marginal product initially increases when more of an input (say labor) is employed, keeping the other input (say capital) constant[34]. Here, labor is the variable input and capital is the fixed input. As more and more of variable input (labor) is employed, marginal product starts to fall. Therefore, following the law, $0 < \beta < 1$. As shown in Figure 4, $\beta = 0.5$. It is just for explanation, otherwise 0.5 might be too harsh in reality. However, with the same computing power, the Sybil approach is more advantageous for reward when $\beta < 1$. On the other hand, monopoly is more advantageous when $\beta > 1$. Since

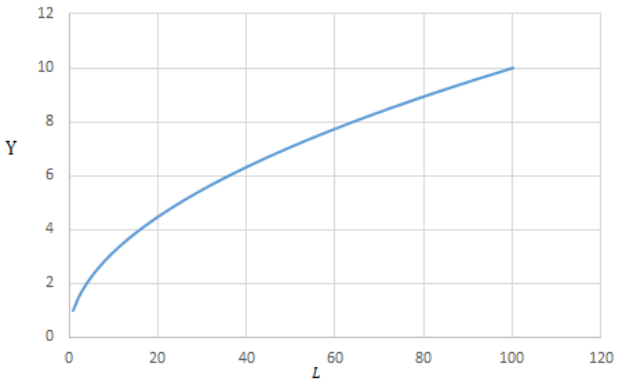

Fig. 4. Production $Y = L^{\beta}$ when $\beta = 0.5$.

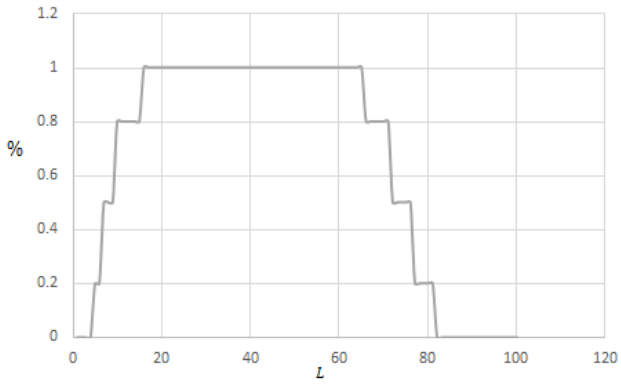

Fig. 5. Mapping table, $L = \bar{m}$.

we can not change the probability to get the reward in hash-based PoW, we can control the grant percentage based on the estimated number of trials. Since the reward in Bitcoin is always as $\beta = 1$, to achieve any β as grant percentage for Bitcoin, we can apply a mapping as $\beta - 1$ for the grant percentage. For example, to achieve final grant percentage as $\beta = 0.5$, we can apply mapping as $\beta = -0.5$ on Bitcoin.

A mapping table as shown in Figure 5 might be further applied to divide the trial range into groups or make the percentage control more sophisticated or artificial, where some organizations might prefer. To discourage monopoly, we can zero the percentage for high trials, or even reject the block to avoid 51% attack directly. On the other hand, to discourage opportunism, we can also zero the percentage for low trials, or even reject the block to avoid dishonesty such as selfish mining. Both grouping to zero above can also reduce the variance of trials estimation. The similar grouping at different levels between 0% and 100% or more smooth curve fitting as in Figure 2 can be applied without significant computation. We can keep the majority in high percentage, not necessarily 100%, to follow the economic laws such as CDPF or diminishing marginal return. Actually, this mapping table plays an important role in customizing blockchains. The final mapping combined with $\beta = 0.5$ and the mapping table as in Figure 5 is shown in Figure 6. Then, we can do scaling and finalize the reward. The scaling can use up the reward budget or save some percentage for other usages, such as taxes or extra bonus.

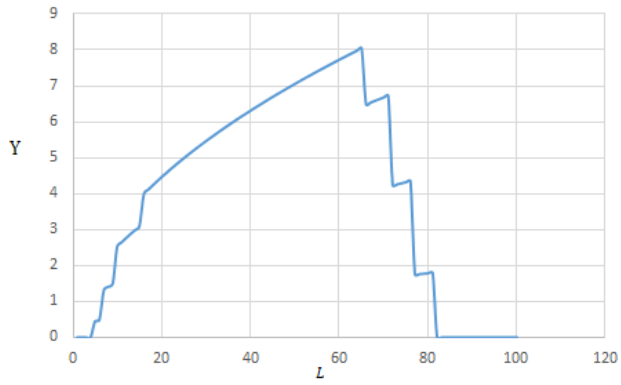

Fig. 6. Mapped final reward.

V. Discussion

To solve the Sybil attack, we can charge some constant fee for all miners by the duration participating in the blockchain. However, if the attacker would like to pay more for the computing power but get less reward, the Sybil attack can only be relaxed economically. To meet the government need, we can tax on the scaled reward for each block. However, to be used in the cyber world, the economist might have better solutions by adjusting traditional economic laws. We believe more sophisticated and feasible mapping functions are available and it will be more reasonable in the cyber world.

As shown in Figure 2, when the nonce trial m is small, the probability of different trial ranges varies a lot. Attacks taking advantage of this so that more than 1 confirmations are needed. As we can reject those blocks with low trials, the trials become more predictable and the number of confirmations needed to avoid attacks, such as double-spending, can be lowered. When we reject those block with high trials, the reward still increases with the computing power, under certain bound. However, it is disadvantageous for miners to increase computing power beyond. Therefore, the difficulty will drop even with the same block interval, and so will power consumption. The computing power can be saved to support other computation-intensive work such as zero-knowledge proof. Mining a block will be easier for the general public except for the ones with very little computing power and block interval can also be shortened. This does not imply higher TPS but encourages miners with minimum computing power to participate. Consequently, more miners can be organized into more big enough groups such as in sharding[10] or Bitcoin-NG[4] to parallelize blockchain for higher TPS indirectly. With the parallelization, in additional to using BFT, scalable storage using PoW can also be practically feasible[15][18] .

Since the probability for a hash-based PoW trial is fixed, lowering difficulty can shorten the block interval but the

average reward might be still the same because block interval might not be in the reward formulas. The number of confirmations might increase, but the block interval might also be shortened. The total confirmation time might be shortened depending on the final reward mapping. Actually, different combinations of mapping tables or even changing dynamically might further prevent the Sybil or other attacks. This could also be a good machine learning topic. *EPoW* makes all these possible, and these can be proved mathematically.

VI. Conclusion

We propose *EPoW*, a simple improvement to hash-based PoW. With an estimable proof-of-work, blockchain problems and attacks might be relaxed or solved economically. The idea can be easily adopted by other consensus protocols. Actually, blockchain is just like a distributed real-time operating system using priority scheduling with an immutable file system, where PoW defines the scheduling priority, consensus protocol does the scheduling, mining is the context switch, block interval is the time slice, and smart contract might be the remote procedure call. Multiple users work on multiple nodes together as a system service to provide trust in a single-process manner, where sharding is to make it multiprocessing. Mining as context switch should not take so much computation power, while transaction processing is the most important real user task. *EPoW* provides information of individual computing power contributed and helps to make this distributed operating system easier to be adjusted, manipulated, and optimized. The ideal scenario is that miners can spend ignorable computing power helping operating the trust machine and the reward is distributed to all participants economically and fairly based on estimable contribution. Using *EPoW*, we believe blockchain and digital currency can be easily customized by companies, organizations or countries for different purposes or policies and be participated by any users in scale.

Acknowledgment

The authors would like to thank Ja-Ling Wu, Yuh-Dauh Lyuu, Wen-Chin Chen, Pangfeng Liu, Jiun-Ming Chen, Yung-Chen Hsieh, Ruey-Long Hong, Vivian Huang, and Leon Hsueh for their different kinds of help; otherwise this work would not be possible. This research was supported in part by grants from the Ministry of Science and Technology, R.O.C. MOST 105-2221-E-002-169- and MOST 105-2218-E-002-017-.

References

[1] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, pp.199-203, 1983.

[2] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology-CRYPTO'92*, pp. 139-147, 1992.

[3] D. Chaum, A. Fiat, N. Naor, "Untraceable electronic cash," in *Advances in Cryptology-CRYPTO'88*, pp. 319-327, 1990.

[4] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: a scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Santa Clara, CA, Mar. 2016.

[5] I. Eyal and E. G. Sirer, "Majority is not enough: bitcoin mining is vulnerable," in *Financial Cryptography*, Christ Church, Barbados, 2014.

[6] J. Garay, A. Kiayias, N. Leonardos, "The bitcoin backbone protocol: analysis and applications," Technical report, 2014.

[7] E. Heilman, A. Kendler, A. Zohar, S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th USENIX Security Symposium*, USENIX Security 15, Washington, D.C., USA, pp. 129-144, Aug. 2015.

[8] G. Karame, E. Androulaki, S. Capkun, "Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin," in *Proc. of Conference on Computer and Communication Security*, 2012.

[9] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, E. Hossain, "Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains," in *IEEE Transactions on Industrial Informatics*, 2017.

[10] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, A secure sharding protocol for open blockchains, in *ACM Conference on Computer and Communications Security*, Vienna, Austria, Oct. 2016.

[11] S. Nakamoto, "Bitcoin: A Peer-to-peer Electronic Cash System," Oct. 2008. https://bitcoin.org/bitcoin.pdf.

[12] C. Natoli and V. Gramoli, "The balance attack against proof-of-work blockchains: The R3 testbed as an example," arXiv preprint arXiv:1612.09426, Dec. 2016. http://arxiv.org/abs/1612.09426.

[13] C. Natoli and V. Gramoli, "The blockchain anomaly," arXiv preprint arXiv:1605.05438, May 2016. http://arxiv.org/abs/1605.05438.

[14] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: generalizing selfish mining and combining with an eclipse attack," *Cryptology ePrint Archive*, Report 2015/796, 2015. http://eprint.iacr.org/2015/796.

[15] R. Padilha and F. Pedone, "Scalable byzantine fault-tolerant storage," in *IEEE/IFIP 41st International Conference on Dependable Systems and Networks*, 2011.

[16] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Cryptology ePrint Archive*, Report 2016/454, Sep. 2016. https://eprint.iacr.org/2016/454.

[17] M. Rosenfeld, "Analysis of hashrate-based double spending," arXiv preprint arXiv:1402.2009, 2014. http:// arxiv.org/abs/1402.2009.

[18] R. Rodrigues and B. Liskov, "Rosebud: a scalable byzantine-fault-tolerant storage architecture," *MIT CSAIL*, Tech. Report TR/932, 2003.

[19] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*, pp. 507-527, 2015.

[20] *Bitcoin*, https://en.wikipedia.org/wiki/Bitcoin, [10- Jun- 2017]

[21] *Bitcoin Source Code*, https://github.com/bitcoin/bitcoin/tree/master/src

[22] *Blockchain*, https://en.wikipedia.org/wiki/Blockchain, [10- Jun- 2017]

[23] *Byzantine fault tolerance*, https://en.wikipedia.org/wiki/Byzantine_fault_tolerance, [10- Jun- 2017]

[24] *Cobb-Douglas production function*, https://en.wikipedia.org/wiki/Cobb%E2%80%93Douglas_production_function, [10- Jun- 2017]

[25] *Blockchain All Time Difficulty*, https://blockchain.info/charts/difficulty?timespan=all, [10- Jun- 2017]

[26] *Blockchain Chart*, https://blockchain.info/charts, [10- Jun- 2017]

[27] S. Deetman (29, Mar, 2016) *Bitcoin Could Consume as Much Electricity as Denmark by 2020*, https://motherboard.vice.com/en_us/article/bitcoin-could-consume-as-much-electricity-as-denmark-by-2020, [10- Jun- 2017]

[28] *Blockchain Could Be Most Significant Innovation Since the Internet: Survey*, http://www.eweek.com/enterprise-apps/blockchain-could-be-most-significant-innovation-since-the-internet-survey, [10- Jun- 2017]

[29] *Bitcoin Energy Consumption Index*, https://digiconomist.net/bitcoin-energy-consumption, [10- Jun- 2017]

[30] *Ethereum mining*, https://github.com/ethereum/wiki/wiki/Mining

[31] *Ethereum*, https://en.wikipedia.org/wiki/Ethereum, [10- Jun- 2017]

[32] *Faulhaber's formula*, https://en.wikipedia.org/wiki/Faulhaber%27s_formula, [10- Jun- 2017]

[33] *Hash function*, https://en.wikipedia.org/wiki/Hash_function

[34] *Marginal product*, https://en.wikipedia.org/wiki/Marginal_product

[35] *Mixing service*, https://en.bitcoin.it/wiki/Mixing_service

[36] *Proof-of-work system*, https://en.wikipedia.org/wiki/Proof-of-work_system, [10- Jun- 2017]

[37] *Proof-of-stake*, https://en.wikipedia.org/wiki/Proof-of-stake

[38] *Transaction Rate*, https://blockchain.info/charts/transactions-per-second

[39] *Scalability*, https://en.bitcoin.it/wiki/Scalability, [10- Jun- 2017]

[40] *Secure Hash Algorithms*, https://en.wikipedia.org/wiki/Secure_Hash_Algorithms, [10- Jun- 2017]

[41] *Smart contract*, https://en.wikipedia.org/wiki/Smart_contract

[42] *What is the GHOST protocol for Ethereum?*, https://github.com/ethereum/wiki/wiki/Mining, [10- Jun- 2017]

[43] *Weaknesses*, https://en.bitcoin.it/wiki/Weaknesses, [10- Jun- 2017]

[44] *Zero-knowledge proof*, https://en.wikipedia.org/wiki/Zero-knowledge_proofs, [10- Jun- 2017]