Towards a More Democratic Mining in Bitcoins

Goutam Paul[1], Pratik Sarkar[2], and Sarbajit Mukherjee[3]

[1] Cryptology and Security Research Unit,
R. C. Bose Centre for Cryptology & Security,
Indian Statistical Institute, Kolkata 700 108, India
goutam.paul@isical.ac.in
[2] Department of Computer Science and Technology,
Indian Institute of Engineering Science and Technology,
Shibpur, Howrah 711 103, India
iampratiksarkar@gmail.com
[3] Department of Computer Science,
Utah State University, Logan, UT, 84322
sab.mukh90@gmail.com

Abstract. Bitcoin is a peer-to-peer electronic cash system that uses a decentralized architecture. It has enjoyed superiority compared to other cyptocurrencies but it has also attracted attackers to take advantage of the possible operational insecurity. All the Bitcoin miners independently try to find the winning block by finding a hash lower than a particular target. On 14^{th} June 2014, a particular mining pool was able to take control of 51% of Bitcoins processing power, thus extracting the maximum amount of profit for their work. In this paper, we introduce a new defense against this 51% attack. We modify the present block header by introducing some extra bytes and utilize the Timestamp more effectively in the hash generation and suggest an alternative to the existing Proof-of-Work scheme. The proposed approach does not rely on finding a hash value lower than the target, rather it awards the miner involved in generating the minimum hash value across the entire distributed network. Fraudulent activities easily get caught due to effective use of the Timestamp. The new scheme thus introduces fair competition among the miners. Moreover, it facilitates the generation of Bitcoins at a fixed rate. Finally, we calculate and show how the new scheme can lead to an energy-efficient Bitcoin.

Keywords: Bitcoins, Electronic Cash System, Miners, Proof-of-Work.

1 Introduction

In Bitcoin, electronic payments are performed by generating transactions that transfer Bitcoin coins (BTCs) among Bitcoin users. Users are referenced in each transaction by means of virtual pseudonyms referred to as Bitcoin addresses. Each address corresponds to a unique public/private key pair. These keys are used to transfer the ownership of BTCs among addresses [1]. Users transfer coins to each other by issuing a transaction [17]. Two types of information are

A. Prakash and R. Shyamasundar (Eds.): ICISS 2014, LNCS 8880, pp. 185–203, 2014.

processed in the Bitcoin system: transactions and blocks [14]. Transfer of value across the system is referred to as transactions, whereas blocks are used to store these transactions and maintain a synchronization among all nodes in the network. A transaction is formed by digitally signing a hash of the previous transaction where the coin was last spent along with the public key of the future owner and finally incorporating the signature in the transaction.

The transactions need to be verified. Any peer can verify the authenticity of a BTC transaction by checking the chain of signatures. Rather than depending on a centralized authority, for this purpose, the Bitcoin system relies on a network of miners who collectively work towards implementing a replicated ledger for keeping track of all the accounts in the system. Each node in the Bitcoin network maintains a replica of this ledger. The replica is constantly updated with time so that the validity of the transactions can be verified against them.

All valid transactions, included in a block, are forwarded to all users in the network to check the correctness of the block by verifying the hash computation. If the block is deemed to be valid, the users append it to their previously accepted blocks. Since each block links to the previously generated block, the Bitcoin block chain grows upon the generation of a new block in the network. Bitcoin relies on this mechanism to resist double-spending attacks. For malicious users to double-spend a BTC, they would not only have to redo all the work required to compute the block where that BTC was spent, but also recompute all the subsequent blocks in the chain [2].

2 Proof-of-Work and Its Weaknesses

In this section, we present a brief description of the Proof-of-Work [8], abbreviated as PoW, and then discuss the various weaknesses associated with the PoW protocol along with some practical examples. In the absence of any centralized payment system, Proof-of-Work is a protocol used to artificially impose transaction costs. The main goal is to "charge" the requester of a service with the efforts to provide a solution to a puzzle, which would be much harder to do than to be verified.

Nakamoto [25] proposed an innovative use of this principle by utilizing it as a core component in the design of a fully decentralized peer-to-peer cryptocurrency called Bitcoin. To prevent double-spending of the same BTC, Bitcoin relies on a hash-based Proof-of-Work (PoW) scheme to generate blocks containing the valid transactions. The goal here is to generate a hash value which must be lesser than a specified target, which is adjusted with time (see Figure 3). The hash basically contains the Merkle hash of all valid and received transactions which the user wants to include in a block, the hash of the previous block, a Timestamp and a nonce value chosen by the user. If such a nonce is found, users then include it along with other entities, that were needed to generate the hash, in a new block and distribute it publicly in the network. Thus the PoW can be publicly verified by other users knows as miners. Upon successful generation of a block, a miner is granted a fixed amount of BTCs, known as coin-based transaction,

plus the transaction fees from all the transactions that have been included in the block. This provides an incentive for users to continuously mine Bitcoins. But still there are some important weaknesses associated with the PoW scheme in Bitcoins.

2.1 Rich Gets Richer, Poor Gets Poorer

Here we identify, how due to the existing protocol, there is an unfair competition among the miners. Bitcoin network purely relies on trustless consensus. Thus if a situation arises when a mining pool controls majority of the voting power, then it could cause havoc.

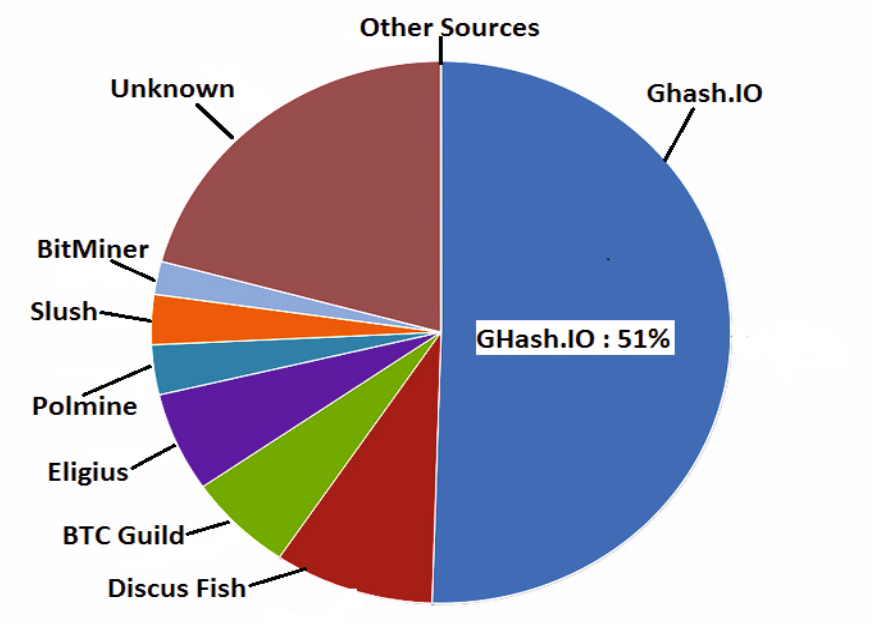

Fig. 1. GHash.io mining pool controlling about 51% of the total processing power (from [18])

A group of miners having 'rich' computational resource may set up a mining pool in such a way that it may control more than 50% of the network's computing power. In such a case that mining pool has the liberty to either modify the ordering or exclude the occurrence of transactions by launching a 51% attack [24]. With the combined mining power, the pool may indulge in double spending by simply reversing transactions that they send. Thus having the required computational power, the pool may be able to validate series of blocks in the block chain by just unscrambling the encrypted series of numbers attached to every Bitcoin transaction. It may also prevent other valid transactions from

being confirmed or reject every block found by competing miners. They cannot directly affect the BTCs stored in the user wallets but they would have the power to make certain addresses unusable. And that allows them to impose any mining fee they like. The mining pool keeps on earning maximum profit and thus the use of the term 'Rich gets richer' sounds appropriate.

On 14^{th} June, 2014, a particular mining pool, namely GHash.io [12,18], was able to take control of 51% of Bitcoins processing power, thus extracting the maximum amount of profit for their work. Figure 1 shows the amount of Bitcoin processing power held by each major mining pool on 14^{th} June, 2014. The 'Unknown' group represents the individual users who are not associated with any mining pools, while others, for example, BitMinter, Eligius etc, are different mining pools associated with solving the PoW puzzle. Thus, it can be easily seen that the pools dominate the process of mining.

2.2 Block Races and Selfish Mining

Another problem that may be associated with the PoW protocol is that of 'race attack'. It can be viewed as an attack originated due to double–spending. Such problems arise from transactions that occur within a short interval of time. Thus it becomes tougher to confirm their verification. On the other hand the the PoW protocol requires time (on an average 10 minutes) to verify a block [22]. So within the verification time a Bitcoin exchange might be completed. In this type of attack, an attacker simultaneously sends an illicit transaction log to the seller and another log to the rest of the peers in the Bitcoin network, where the original owner gets back his currency. But by the time the seller realizes that he has received a fraudulent amount, the transaction may have already been carried out.

'Selfish Mining' can also be another possible attack. It was first introduced by Ittay Eyal and Emin Gun Sirer in [20] In this attack, when a miner solves the PoW puzzle and verifies a new block, he keeps it with himself. Thus by not distributing it over the network, he doesn't allow others to work on the next block. Instead, the miner starts working on the next puzzle that would verify the block which would follow his unreleased block. Thus if a mining pool is set up, they might use their overall computational power to keep verifying blocks. Finally when other miners find a new fair-mined block, the selfish miners releases their verified chain of blocks, which might be of several blocks. Their blocks would automatically be added to the main Bitcoin chain and the selfish miners would always gain, since the longer chain always wins. The rest of the miners didn't have the notion of those hidden blocks and that resulted in wastage of hashing power.

2.3 Illegal Usage of Machines for Mining

In Bitcoin mining, an algorithm or a puzzle is needed to be solved, that has increasing complexity related to the number of Bitcoins in circulation. Attackers try to exploit this mechanism of mining by illegally using computational

resources, for example, by infecting a huge number of machines [21] in the network with malware, thus building a malicious botnet, that would be able to mine Bitcoins. The attack may be executed by a fake and infected version of legitimate software, packaged with malware or it may happen while clicking on malicious shortened URLs spammed through Email or social Media platforms. Once infected, the computational resources of the victim are used in the mining process.

Another form of attack which recently occurred was the illegal use of supercomputers of research groups for mining Bitcoins. It was first reported in Harvard [28], where a mining operation had been set up on the Odyssey cluster of the Harvard research network. Similar incidents occurred in Imperial College of London [27] and in the research labs of USA [10].

2.4 Wastage of Computing Power

The PoW problem is generally solved using ASIC machines which have been specially designed for this purpose.

Each PoW problem generally requires 10^8 GH/s (Gigahashes/second) to be solved, which can be seen from the chart in Figure 2. Such enormous requirement of computing power attracts illegal use of supercomputers and large computing power machines for Bitcoin mining.

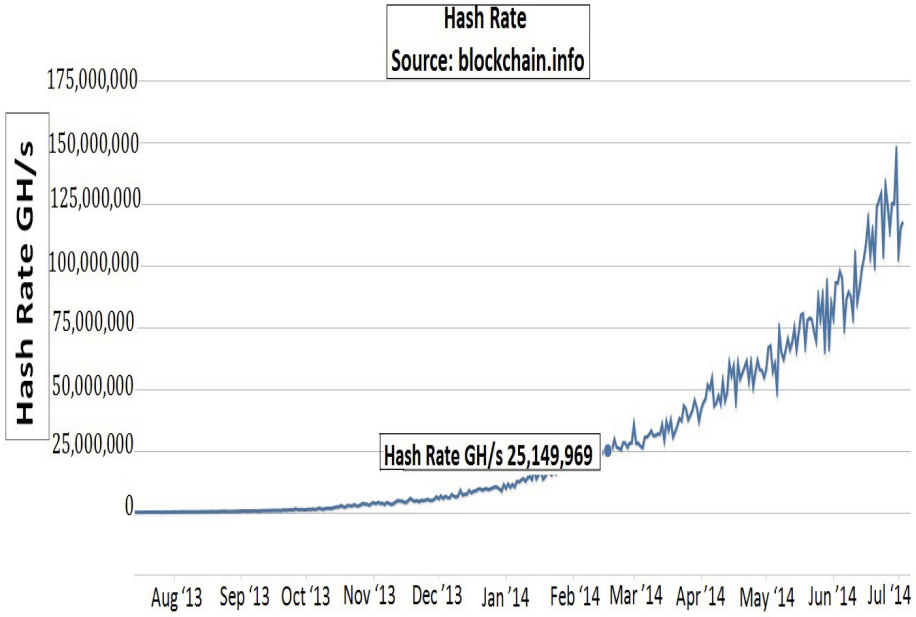

Fig. 2. Chart showing the hash-rate required in solving the PoW puzzle (from [16])

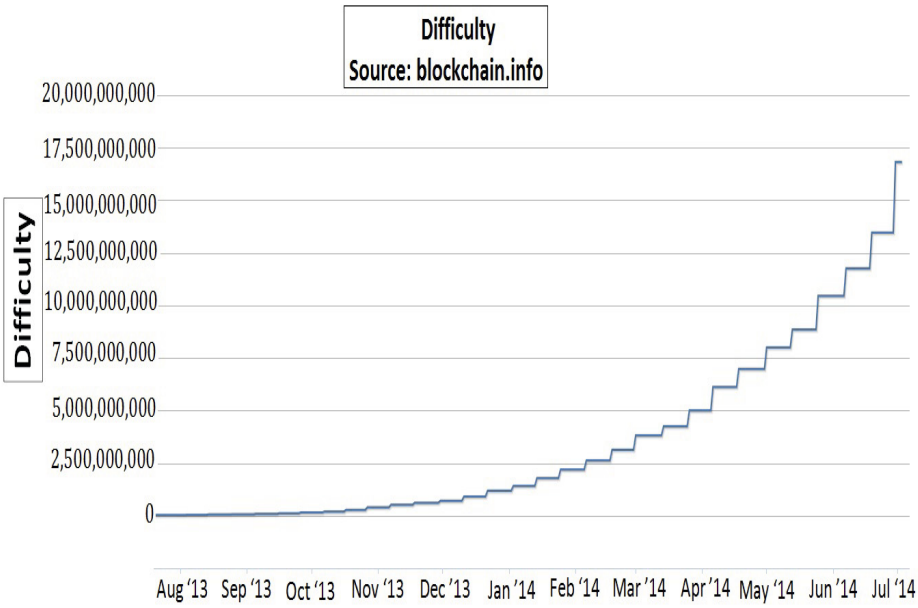

Fig. 3. Graph showing the variation of difficulty in the system (from [15])

According to "Bitcoin Watch" [23], the whole Bitcoin network hit a record-breaking high of 1 exaFLOPS a year earlier. FLOPS would basically mean the number of Floating-point Operations a computer can do Per Second, or how fast it can solve math problems. An exaFLOPS is 10^{18} math problems per second. The most powerful supercomputer in the world, Sequoia, can manage a mere 16 petaFLOPS, or just 1.6 percent of the power geeks around the world have brought to bear on mining Bitcoin. The world's top 10 supercomputers can muster 5 percent of that total, and even the top 500 can only muster a mere 12.8 percent. The new ASIC machines used by the miners are built from scratch and are only used to mine Bitcoins. Thus they can't serve any other purpose. So the total power spent on Bitcoin mining could theoretically be spent on something else, like real world problems that exist naturally. From Figure 2 and Figure 3, we see that the Hash-rate and Difficulty level is exponentially increasing. This would require more computation power for solving the PoW puzzle and in turn would waste more computing power in the future.

2.5 No Guarantee of Coin Generation at a Fixed Rate

The most important aspect of this discussion is the rate at which the Bitcoins are generated. The graph in [13] shows that the time required for confirmation of a transaction usually takes around 5-20 minutes, which is against the policy

where each transaction verification should take on an average 10 minutes. If more miners join in the race to find the puzzle for verifying the block, more hashes would be generated and tested within the same time-span. But according to the Bitcoin protocol, the network self-regulates the speed of generation of Bitcoins after a certain time-span (after every 2016 blocks) by checking the number of days required to generate x many hashes. If the time-span is found out to be too short, then the difficulty level of PoW puzzle is increased. and so it becomes harder to find out the required hash in the next round. Thus if a mining pool is set up which could control more than 50% of the computational power, then such rules can be broken.

The problem lies in the self–regulation of the network. If the network members detect that the last hash generation took too much time, then the difficulty level will be adjusted down, and in the next round, the hashes will be easier to be found out. This actually means that if all peers in the network agree on only using 1% of their available computing power, thus also only 1% of electricity and 1% of the electricity bill they have right now, the entire Bitcoin system would still continue to work exactly as before. Everybody would still get the same payout in amount of Bitcoins received, and the self-regulating nature of the network means that exactly the same total amount of Bitcoins would be generated as before. But the fees involved in mining has increased the greed of the miners and they are wasting more and more computing power for mining which results in the exponential rise of the target difficulty. The graph in Figure 3 shows how the target difficulty is managed by the system.

3 Existing Alternative Proposals and Their Disadvantages

In this section, we provide a brief discussion of the alternative strategies that have been proposed to tackle the weakness associated with the PoW protocol. We also point out some weaknesses associated with those schemes.

3.1 Proof-of-Burn

The idea of proof-of-burn [7] is that the miner should show that they have burned some coins in order to generate new coins. Coins can be burnt by sending them to verifiable unspendable addresses. Only those users who have burnt coins in the past can generate new coins in future and gain the transaction and block generation fees. The metric for mining coins is the burning of coins. This system successfully reduces the computing power for mining but it also wastes Bitcoins in the process.

3.2 Proof-of-Stake

The proof-of-stake concept was first introduced by an user QuantumMechanic in [11]. Each active user can show his stake in the system by proving the number of Bitcoins held by him in his addresses. The larger the number of Bitcoins held

by an user, the larger will be his stake. Each user holding $y\%$ of stake can mine only $y\%$ of the proof-of-stake coin. Proof-of-stake has been used in Peercoin [29]. As this concept is based on the amount of stake held by an user, it poses a threat of centralization. If an user gains more than 50% of the system Bitcoins then he can monopolize the system and perform double spending or deny service to rest of the users. Thus, the number of Bitcoins held by an user cannot be used as a metric for the Bitcoin security system.

4 Our Proposal

In this section, we describe our proposed scheme. We first describe in short the various resources needed to present our algorithm. Section 4.2 provides an in-depth description of our proposed algorithm.

4.1 Resources Needed

Peer-to-Peer Network. The Bitcoin users are connected in a peer to peer network [6]. They broadcast the details of transactions and blocks over the network using TCP communication.

Timestamp Server. The original Bitcoin paper [25] proposed the idea of a Timestamp server. After that the Bitcoin blockchain contains Timestamped transactions and blocks. This Timestamp helps to prove the existence of the data.

Bitcoin Address. A Bitcoin user has an account, where each account is associated with one or more Bitcoin addresses [1]. A Bitcoin address is an 27-43 length alphanumeric string that represents an address for payments of Bitcoins. Each address has a corresponding private key which is required in order to spend the coins at that address. The address and the private key is a pair of ECDSA [4] keys, where the address is the public key and the private key of the address is the private ECDSA key.

4.2 Description of Our Algorithm

Each user generates a hash, based on which the miner of the next block is decided. The user whose hash is of minimum value amongst all the users in the system, will generate the next block. He will receive the transaction fees for all the transactions that are verified in his block and it will be added to the blockchain as the next block. He will also initiate a coinbase transaction to his public address and award himself with a specified amount of Bitcoins. This award to himself is an additional incentive for verifying the blocks. This scheme deals with the algorithm for generation, broadcast and verification of the hash. It is divided into 3 phases, *hash generation phase, hash broadcast phase* and *hash verification stage*. These 3 phases together run for 10 minutes and give the true

minimum hash of the system. We call this duration as the *time frame* which is maintained by the Bitcoin system time. By true minimum, we mean the actual minimum hash amongst all the hashes that has been generated by the miners, and broadcasted in the system. At the end of the algorithm, the miner who has generated the true minimum hash will form the block.

Phase 1: Hash Generation Phase. In this phase the user/miner verifies the transactions and forms a Merkle root tree [5] from those transactions. He generates a block from those transactions with a block header as described in Table 2. He computes the minimum possible hash from the block header by changing the value of the Nonce. This phase continues for 2 minutes and this time is maintained by the Timestamp T (refer to section 4.1) in the hash message. Any message that has been generated after this 2 minutes will be discarded.

Table 1. Present Block Header used in the Bitcoin network. (from [3])

Name	Byte Size	Description
Version	4	Block Version Number.
Previous Hash	32	This is the hash of the previous block header.
Merkle Root	32	The hash based on all the transactions present in the current block.
Time	4	Current Timestamp in seconds (unix format).
Target Bits	4	Target value in compact form.
Nonce	4	User adjusted value starting from 0.

Table 2. Proposed Block Header in our design

Name	Byte Size	Description
Version	4	Block Version Number.
Previous Hash	32	This is the hash of the previous block header.
Merkle Root	32	The hash based on all the transactions present in the current block.
Time	4	Current Timestamp in seconds (unix format).
Bitcoin Address	20	Hash of the Public key of the receiving address.
Nonce	4	User adjusted value starting from 0.

Block Header: We recommend to modify the present 80 bytes block header (Table 1) of the Bitcoin block structure and replace it with the newly proposed block header of 96 bytes (Table 2). The user uses 5 fields similar to the previous block header and includes an additional field (Table 3), i.e. his own public address that is currently being used. The 4 bytes 'Target Bits' field in the header is not required, since in the newly proposed design there is no specific target for the miner to fulfill. The public address of the user which is used during the hash generation, is required to compute the hash. The scheme inserts a new 20 bytes of Bitcoin address U_p field to the block header.

The hash (H) is generated from the 7 fields mentioned in Table 3.

Table 3. Items used by the User to generate the Hash

Version (V)	Previous Block Hash (H_p)	Timestamp (T)	Bitcoin Address (U_p)	Hash of Merkle Tree of verified transactions (H_t)	Nonce (R)	Padding (P)

1. *Version (V)* - The first 4-bytes of the block header describes the block version number V used by the miner. It is the Block version information, based upon the software version creating this block. In our scheme, we use the similar idea of Version number as given in [3] and [9].
2. *Previous Block Hash (H_p)* - In the Bitcoin system, the blocks in the blockchain are chained to each other where each block contains the hash of the previous block. In the proposed scheme, the chaining system has been retained in order to track the previous block. The previous block hash is denoted by H_p and is of 32 bytes size. In our scheme, Previous Block Hash is used in the same sense as given in [3] and [9].
3. *Timestamp (T)* - It denotes the 4-bytes Timestamp of the starting of hash generation by the user. It is updated after every second and is denoted by T. It is used to check whether a user has broadcasted any hash that has been generated before the hash generation period started or after it has expired. If such fraud has occurred, then the hash can be discarded, if its T field value does not lie within the 2 minutes of the generation phase. The timestamp for the start of the hash generation phase is updated every 10 minutes by the system and is available publicly in the Bitcoin network. So fraudulent activity concerning the starting timestamp for the hash generation phase is not taken into account.
4. *Bitcoin Address (U_p)* - The Bitcoin address U_p is the 160-bit (20 bytes) hash of the public portion of a public/private ECDSA key-pair. In our scheme, we use the similar Bitcoin Address as originally proposed by Nakamoto [25]. A key-pair is generated for each address which will be used by the user for mining and transaction purposes. The address helps to track the user, who has the minimum hash value and grants him the right to form and append the next block in the block chain. A more detailed description of this address can be found in 'Address' page of Bitcoin Wiki [1].
5. *Hash of Merkle Tree of verified transactions (H_t)* - The user verifies all the transactions, that occurred in the last 10 mins and are available in his pool. He forms a Merkle tree [5] from the transactions that have been verified. The 32 bytes root (H_t) of the Merkle tree is used for hash generation, since the broadcasted hash (H_t) of the root of Merkle Tree is a proof that the user has not changed any of the transactions that he verified during the hash generation phase.
6. *Nonce (R)* - The user chooses a 4 bytes nonce R which will help him to generate the hash. In the hash calculation, the 4 fields - H_p, T, U_p and H_t are fixed for a particular user at any time during hash generation, and hence cannot be modified by the user to manipulate and generate different hashes.

So the user can only alter the value of R to generate various hashes, within the hash generation period. He keeps a record of the hashes that he has generated within this time period. He then broadcasts the minimum among the generated hashes to the network after the hash generation period expires and hopes his generated hash turns out to be the minimum among all the hashes submitted by the active users in the Bitcoin network. Thus in this proposed scheme, the minimum hash generation is purely based on ***luck***.

7. *Padding (P)* - It involves addition of some extra bits at the end of the input concatenated string, in order to make it 128 bytes and then it can be divided into two blocks of 64 bytes each.
   The total length T_L of concatenation fields is given by:

$$T_L = 4\ (V) + 32\ (H_p) + 32\ (T) + 4\ (U_p) + 20\ (H_t) + 4\ (R) = 96\ bytes. \quad (1)$$

   Thus, the number of bytes required for padding is:

$$P = 128 - 96 = 32\ bytes. \quad (2)$$

The hash H is a 32 bytes SHA-256 hash of the 128 bytes string containing the concatenation of the above fields. In our scheme, we use the similar hash function as originally proposed by Nakamoto [25] [3]. The equation can be stated as:

$$H = SHA(SHA(V||H_p||T||U_p||H_t||R||P)). \quad (3)$$

***Hash Message*:** Each individual miner will generate a hash message M which will contain two parts - block header and hash H. The hash will be compared and the block header can be used for the verification of the hash. Each miner/node will generate his own message during the generation phase and broadcast it to the network.

For completeness and ease of reference, we provide the algorithm [26] here in our notation. Let, N_i denotes a variable which represents the i^{th} node in the network. The value of i varies from 1 to the total number of nodes connected in the network. E.g. N_1 denotes the first node and so on. N_i contains two fields M_{min} and STATE. M_{min} contains the message with the minimum hash value. M_{min} may be generated by the current node N_i or it may have been received by the node from its neighbors. STATE field shows the current state of the node. The state can vary between AVAILABLE, ACTIVE, PROCESSING or SATURATED. Initially all the nodes are AVAILABLE. They become ACTIVE when they successfully generate a hash at the end of the hash generation phase, and storing that hash in their M_{min} field by calling the 'Initialize(M)' function, where M denotes the hash message generated by the node. All the nodes simultaneously call the Initialize() function with their own hash messages as shown in Algorithm 1. Hence, at the end of the generation phase, each node is in ACTIVE state and its M_{min} field contains the hash message generated by it.

Once the *Hash Generation* phase is over and each node is ready with its message M, the next 8 minutes is used for

```
Procedure Initialize (Hash Message M);
for all Nodes in the network do
    N_i.M_min = M, where M_min is the minimum hash message at each node;
    N_i.STATE = ACTIVE;
end for
```

Algorithm 1. Hash Message Generation Phase at each node in the network

- broadcasting the messages containing the hashes,
- finding the minimum hash amongst all,
- verifying the minimum hash selected,
- detecting dishonest behaviors in the system which involves checking the Timestamp (T) value of the generated hashes.
- granting the mining rights of the next block to the rightful user, who generated the minimum hash.

Phase 2: Hash Broadcast Phase. After the hash has been generated, it is broadcasted to all the active nodes in the network. Now among those hashes, the minimum needs to be found out in the distributed environment. We propose to use a modified version of the distributed algorithm [26] for finding out the minimum.

In the hash broadcast phase, the hash value at each node would be broadcasted out and the minimum hash message value would be found out in the system. Each active leaf starts the broadcasting stage by calling LeafSending() function and sending its M_{min} value to its only neighbor, referred now as its 'parent', and becomes PROCESSING (Note: messages will start arriving within finite time to the internal nodes). The internal node calls the Receiving_Active(M) function on receiving a message M from its neighbors. It calls Process_Message(M) to compute the minimum among the current minimum hash value it is holding and the one that it has received and stored it in M_{min}. It waits until it has received a message M from all its neighbors but one, and then sends its M_{min} message to that neighbor that will now be considered as its 'parent' and becomes PROCESSING. If a PROCESSING node receives a message from its parent, it calls Receiving_Processing(M) function and becomes SATURATED. Algorithm 2 describes the above procedure.

The algorithm implies that (Lemma 2.6.1 in [26]) exactly two PROCESSING nodes will become SATURATED; furthermore, these two nodes are neighbors and are each others' parent. All the other nodes will be in the PROCESSING state. They will start the hash verification stage by making their minimum hash message public. If there has been any discrepancy in hash broadcasting by dishonest nodes, then it will be resolved in hash verification stage. Other nodes will understand that the hash verification stage has started when they will see that one message has been publicly broadcasted. We present the hash comparison method that has been used in this phase.

```
————————ACTIVE————————-

Procedure LeafSending()
for all Active Leaf Nodes in the network do
   parent ⇐ Neighbors;
   send N_i.M_min to parent;
   N_i.STATE = PROCESSING;
end for

Procedure Receiving_Active(M)
for all Active Internal Nodes in the network do
   N_i.M_min = Process_Message(M);
   Neighbors:= Neighbors - sender;
   if number of Neighbors = 1 then
      parent ⇐ Neighbors;
      send N_i.M_min to parent;
      N_i.STATE = PROCESSING;
   end if
end for

————————PROCESSING————————-

Procedure Receiving_Processing(M)
for all Processing Nodes in the network do
   N_i.STATE = SATURATED;
   N_i.M_min = Process_Message(M);
   Announce M;
   Start Verification stage;
end for

————————————————————

Procedure Process_Message(M)
for all Nodes in the network do
   if N_i.M_min.H < M.H then
      return N_i.M_min;
   else
      return M ;
   end if
end for
```

Algorithm 2. Distributed Algorithm for computing minimum hash value

Hash Comparison: Two hashes are compared from the most significant bit (MSB) to the least significant bit (LSB). They are compared from left to right until one bit differs among them. The one with the lower changed bit is the smaller hash. If two hashes are of same value then the header message containing the lower Timestamp will be considered the minimum one. Let us consider an example. Here we have two 32-bytes hashes H_1 and H_2 in hexadecimal format.
$H_1 : 1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64,$
$H_2 : 1312afaf42fc31103f1fdc0151fa7471187349a4714df7cc11ea464e12dcd4e9.$

The first 3 bytes (underlined) of both the hashes are same. But the 4^{th} bytes are different. The 4^{th} byte (bold) of H_2 is bigger in hexadecimal format, so H_1 is considered as the smaller hash.
$H_1 : \underline{1312af}\mathbf{17}8c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64,$
$H_2 : \underline{1312af}\mathbf{af}42fc31103f1fdc0151fa7471187349a4714df7cc11ea464e12dcd4e9.$
We have $H_1 < H_2$, since $17 < af$.

Phase 3: Hash Verification Phase. It is possible that the true minimum hash might not have been broadcasted properly by dishonest nodes and hence a hash value, bigger than the true minimum value is chosen at the end of the hash broadcast phase. Thus, the hash verification stage is required to verify, and if necessary, find the true minimum hash of the system. The message that has been publicly broadcasted by the two saturated nodes will be verified by the others nodes. If any node has a hash value lesser than the hash of that broadcasted message, then he can claim for his hash. His message will also be verified by the other nodes. If the new minimum hash value message is found to be legitimate, then its corresponding user will be the winner of the block. If the hash broadcast and verification stage is completed in less than 8 minutes then the miner will form the block and the system will wait for the 8 minutes period to expire until it can again allow the mining of a new block. The T value will check that miners cannot mine during this extra period, to prevent any unnecessary advantage.

Block formation: The user who has been identified as the generator of the minimum hash will form the block. He will incorporate the transactions in the block and the block header into the block chain. The user being the miner of the block will also initiate a coinbase transaction in the block in order to generate new Bitcoins and award himself with those coins as well as with the transaction fees from each verified transaction.

4.3 Message Complexity

The total number of active nodes that are participating in the mining process is denoted by n. We utilized the saturation stage broadcasting [26] for our minimum hash finding. During the hash broadcast phase, exactly one message is transmitted on each edge, except the two saturated nodes (from equation 2.24 of [26]). The two saturated nodes exchange two messages. So the total number of messages transmitted are:

$$n - 1 + 1 = n. \tag{4}$$

Thus, the message complexity of the scheme is $O(n)$. The time frame of 8 minutes of hash broadcasting can be easily increased by allowing the Bitcoin system to change the time frame. Increasing the time frame will accommodate more users to join in the mining process in the future and still allow generation of Bitcoins at a fixed rate.

4.4 Security Issues

In this section, we discuss salient security features of our scheme.

The node with the highest computing power in the system does not gain any advantage on other nodes for generating the lowest hash of the system. Each node is unaware of the hash generated by the other active nodes in the system. The value of the timestamp changes every second, so the nodes cannot manipulate the result of the hash after 2 minutes by only manipulating the nonce. If two nodes generate the same lowest hash, then the node whose timestamp value is less will win, i.e., the node who generated the lowest hash first will win. Thus, nodes with larger computing power can generate more hashes in those 2 minutes but it does not guarantee their win.

The verification of the hashes is performed using the original Bitcoin procedure of verifying newly mined blocks. Even if the true minimum hash of the system may not be transmitted by the dishonest nodes during broadcasting, it can be claimed and then verified by the peer nodes during the hash verification stage. The dishonest nodes may dominate the system, but during verification, the hash and the block header is made public. A hash will be discarded only if the hash value does not match with the hash of the header fields or it is bigger than some other hash, which has been verified. So evil nodes cannot affect the verification stage.

5 Comparison with Proof-of-Work Protocol and Its Existing Alternatives

In this section, we try to draw a comparison between the original PoW scheme and our proposed scheme on the basis of certain factors:

- Message Complexity of the scheme,
- Primary Concern of the scheme,
- Possibility of the 51% Majority Attack,
- Hash Rate (per miner),
- Competition among miners to generate a new block,
- Time to generate.

We also try to compare our existing scheme with proof-of-stake protocol that has been implemented in new type of Cryptocurrency known as PeerCoin [29]. The comparison has been shown in Table 4.

From the comparison shown in the table, the superiority of our scheme lies in the fact that it defends well against the 51% Majority Attack, which is currently the primary concern among the Bitcoin community. It also provides a fair

Table 4. Comparison between Proof-of-Work, Proof-of-Stake and Our Scheme

	Message Complexity	**Main Concern**	**51% Attack Possible ?**	**Hash Rate (per miner)**	**Competition**	**Time to generate**
Proof of Work	Only one miner is chosen as winner. So low message complexity	Large Computations required	Yes	Very Large	Unfair competition among miners	Variable time
Proof of Stake	Every miner involved (having stakes) in exchanging messages. High message complexity	Initial distribution of stakes	Yes	Large	Unfair competition among stake holders	Variable time
Our Scheme	Every miner involved in exchanging messages. High message complexity	Flooding of messages in system	No	Small	Fair, competition purely based on luck	10 mins

competition among the miners, since the generation of block in this scheme is purely based on luck. It will not matter if some miner or some mining pool has machines with very high computation power capable of solving large problems within a few seconds. All that matters, is the generation of the minimum hash in the network among all miners. No miner would know what hash value other miners are generating. Even if they try to modify their generated hash value by observing the hash values broadcasted in the network, they would eventually get caught due to the presence of the Timestamp field in the block header. Since they would be considered to be fraud by generating a hash value beyond the Hash generation phase. Thus this scheme is purely based on luck and proves its effectiveness against the primary weakness of PoW protocol.

6 Towards Greener Bitcoins

The PoW protocol uses ASIC machines for mining and hence consumes a lot of energy during the process. Here we show that our scheme reduces the energy consumption by at least 5 times than that of the PoW protocol, using the same ASIC machines. Table 5 from Bitcoin Wiki [2] shows the hash power and energy usage of the machines. The PoW protocol requires these machines to meet the target, else generation would require more than 10 minutes. Also we have shown earlier in Figure 3 that the difficulty level is rising gradually, thus increasing the hashing power requirement.

Let us give an approximate calculation of the energy usage by these ASIC machines. We are stating an example of the average energy usage by these machines from one of the reputed sources of Bitcoin news [19]. The Bitcoin network has an

Table 5. Bitcoin Mining Hardware Comparison (from [2])

ASIC Unit	Hash Power (GH/s)	Energy Usage (kW)	W / GH	Unit Price
Cointerra TerraMiner IV	2000	2200	1.10000	$5,999
KnC Neptune	3000	2200	0.73333	$9,995
Hashcoins Zeus	3500	2400	0.68571	$10,999
Extolabs EX1	3600	1900	0.52778	$9,499
Minerscube	15000	2475	0.16500	$9,225

average hash-rate of 110 million GH/s. The average energy efficiency has been assumed to be 0.7333 W/GH. The network needs 0.7333 x 110 million Watts = 80,666 kW per hour. This equates to 80,666 kW x 24 hrs/day x 365.25 days/year = 707,120,500 kWh/year.

Thus the PoW protocol requires, on an average, 2.54 million GJ/year. Our protocol generates hashes for 2 minutes in an interval of 10 minutes. So only one-fifth of the total time is required for hashing. Assuming the same network hash-rate of 110 million, the network needs 80,666/5 = 16,133 kW. Our scheme consumes energy of the order of 16,133 kW x 24hrs/day x 365.25 days/year = 141,423,631 kWh/year= 0.508 million GJ/year. Since the scheme is totally uniform for different miners and does not require to meet any computation-intensive target, it can use lower hash rate machines for coin generation which will further reduce energy consumption. Thus, it is a greener approach than PoW.

7 Conclusion

In this paper, we have analyzed the major weaknesses of the existing Proof-of-Work protocol of Bitcoins and proposed an alternative solution. Thus in the proposed scheme, having a large computing power doesn't essentially mean that the user has an upper hand in generating the next block. The block generation is now purely based on luck, where the miner having the minimum hash value in the system during a span of 10 minutes, would be declared as winner. The effective use of Timestamp during the hash generation phase, where only a couple of minutes are allowed for hash generation, is shown to eliminate the chances of any fraudulent activities in the system. It has removed the difficulty target which will allow the miners to generate new Bitcoins using less computing power thus mining in a more environment friendly way. This new scheme generates the coins at a fixed rate, which has not been addressed by any other methods, even though it is one of the fundamental requirements for the Bitcoin system.

Acknowledgments. We thank the anonymous reviewers whose feedback helped in improvement of the technical as well as the editorial quality of our paper. We are also grateful to the Project CoEC (Centre of Excellence in Cryptology), Indian Statistical Institute, Kolkata, funded by the Government of India, for partial support towards this project.

References

1. Bitcoin–Wiki. Address, https://en.bitcoin.it/wiki/Address
2. Bitcoin–Wiki. Bitcoin Wikipedia, https://en.bitcoin.it/wiki/Main_Page
3. Bitcoin–Wiki. Block hashing algorithm, https://en.bitcoin.it/wiki/Block_hashing_algorithm
4. Bitcoin–Wiki. ECDSA, https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm
5. Bitcoin–Wiki. Merkle Tree, https://en.bitcoin.it/wiki/Protocol_specification#Merkle_Trees
6. Bitcoin–Wiki. Network, https://en.bitcoin.it/wiki/Network
7. Bitcoin–Wiki. Proof of Burn, https://en.bitcoin.it/wiki/Proof_of_burn
8. Bitcoin–Wiki. Proof of Work Protocol, https://en.bitcoin.it/wiki/Proof_of_work
9. Bitcoin–Wiki. Protocol specification, https://en.bitcoin.it/wiki/Protocol_specification
10. Bitcoin Magazine. Government bans Professor mining bitcoin supercomputer, http://bitcoinmagazine.com/13774/government-bans-professor-mining-bitcoin-supercomputer/
11. Bitcoin Talk. Proof of stake instead of proof of work, https://bitcointalk.org/index.php?topic=27787.0
12. Bitcoinx. Mining pool giant GHash.io reaches 50% of bitcoin hashing power, http://www.bitcoinx.com/mining-pool-giant-ghash-io-reaches-50-of-bitcoin-hashing-power/
13. Blockchain. Average Transaction Confirmation Time, http://blockchain.info/charts/avg-confirmation-time?timespan=2year&showDataPoints=false&daysAverageString=1&show_header=true&scale=0&address=
14. Blockchain. Blocks, https://en.bitcoin.it/wiki/Block
15. Blockchain. Difficulty, http://blockchain.info/charts/difficulty?timespan=1year&showDataPoints=false&daysAverageString=1&show_header=true&scale=0&address=
16. Blockchain. Hash Rate, http://blockchain.info/charts/hash-rate?timespan=1year&showDataPoints=false&daysAverageString=1&show_header=true&scale=0&address=
17. Blockchain. Transactions, https://en.bitcoin.it/wiki/Transaction
18. Business Insider. Today, Bitcoin's Doomsday Scenario Arrived, http://www.businessinsider.in/Today-Bitcoins-Doomsday-Scenario-Arrived/articleshow/36516972.cms#ixzz34amw9VI2
19. CoinDesk. Under the Microscope: Economic and Environmental Costs of Bitcoin Mining, http://www.coindesk.com/microscope-economic-environmental-costs-bitcoin-mining/
20. Eyal, Ittay and Sirer, Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. CoRR, abs/1311.0243 (2013)
21. Forbes. Brilliant But Evil: Gaming Company Fined $1 Million For Secretly Using Players' Computers To Mine Bitcoin. http://www.forbes.com/sites/kashmirhill/2013/11/19/brilliant-but-evil-gaming-company-turned-players-computers-into-unwitting-bitcoin-mining-slaves/

22. Frequently Asked Questions. Transactions, https://bitcoin.org/en/faq#why-do-i-have-to-wait-10-minutes.
23. Gizmodo. The World's Most Powerful Computer Network Is Being Wasted on Bitcoin, http://gizmodo.com/the-worlds-most-powerful-computer-network-is-being-was-504503726
24. Learn Cryptography. 51% Attack, http://learncryptography.com/51-attack/
25. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (May 2009)
26. Santoro, N.: Design and Analysis of Distributed Algorithms. Wiley Series on Parallel and Distributed Computing, pp. 71–76. Wiley Interscience (2006)
27. The Guardian. Student uses university computers to mine Dogecoin, http://www.theguardian.com/technology/2014/mar/04/dogecoin-bitcoin-imperial-college-student-mine
28. The Harvard Crimson. Harvard Research Computing Resources Misused for Dogecoin Mining Operation, http://www.thecrimson.com/article/2014/2/20/harvard-odyssey-dogecoin/
29. Wikipedia. Proof-of-stake — Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Proof-of-stake&oldid=615023202