An Analysis of the Hash-Based Proof-of-Work Chain in the Bitcoin Network

Neel Gupte

Abstract—A Bitcoin Is an electronic payment method that is based on cryptographic proof instead of trust [1] allowing any two users to manage their transactions without dependence on a third party organization (a financial institution). Each party can transfer Bitcoins to the other digitally by signing a previous hash along with the public key of the next owner and so on. The Bitcoin network timestamps the transactions into blocks and hashes them into a chain of a hash based proof-of-work [1] (block chain) and combining them into a block. Such a chain is known as a block chain. It is decided that the longest block chain shall serve as the proof of the sequence of transactions witnessed by the network. Also it is a proof that it came from the largest pool of computational power put in by Bitcoin Miners (people who help in generating new blocks for the proof-of-work in return for a bounty). A major issue with the Bitcoin network is the problem of attacking nodes trying to double-spend the Bitcoin transactions i.e. re-spend the money they have already spent. They can do so by trying to generate an alternate chain faster than the honest nodes. As long as these honest nodes manage the majority of computational power, the attacker node wont be able to interfere with the block chain. In this paper we see how effective the traditional bitcoin proof-of-work policy is against the attackers and what measures and methods can be adopted in order to nullify the probability of an attacker to interfere with the block chain.

Index Terms—Bitcoin, Cryptocurreny, Block Chain, Hashing, Proof-of-Work, Double-spending, Momentum Method, Proof of Stake.

1. INTRODUCTION

A peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another, without the interference of a financial institution. Though digital signatures provide a solution, main benefits are lost if a trusted third party is still required to prevent double spending. Hence we propose a solution to double spending using a peer-to-peer network. The networks puts timestamps on the transactions by hashing them into an ongoing chain of hash based proof-of-work and combines these transactions into block. The longest chain of these blocks i.e. the longest block chain serves as a proof of the sequence of events/transactions witnessed, but also it's a proof that it came from the largest pool of computational power. As long as Bitcoin Miners (people who help in generating new blocks for the block chain in return for a bounty) i.e. the non-attacking nodes control the majority of the CPU power, they will end up generating the longest and valid block chain, which would get accepted in the network, thus preventing the attacking nodes to modify the block chain. The network itself requires a very minimal structure. Messages are broadcast on a best effort basis and the nodes are allowed to leave the network and then rejoin, accepting the longest and most valid block chain as the proof-of-work as to what transactions took place in the network while they were gone. Bitcoin is an electronic payment system which is based on cryptographic proof instead of trust thus allowing two users to handle transactions with each other without depending on a third party. Each user transfers the Bitcoins to the other user using digitally by signing a previous hash along with public key of the next owner and so on.

2. TRADITIONAL PROOF-OF-WORK

In order for the attacking node to interfere with the block chain, it would have to simultaneously generate an alternate chain of the hash based proof-of-work faster than the actual block chain so that the attackers chain would be accepted as the longest and treated as the new proof-of-work. Once the attacker achieves this, his altered and faulty transactions are accepted into the Bitcoin network allowing him to re-spend any Bitcoins he has already spent. We consider the probability of an attacker trying to generate an alternate chain faster than the honest nodes In-spite of the attacker being successful; the system does not open up to arbitrary changes or any kind of attacks. This is possible because a node will not accept an invalid transaction as payment and no honest node will accept a block with it. Hence an attacking node can only change his transactions in order to re-spend the money he has already spent. The probability of an attacking node to catch up with the block chain is analogous to a Gamblers Ruin problem. We can calculate the probability that a gambler with unlimited credit plays an infinite number of trials to reach breakeven or that an attacker catches up with an honest block chain. [2]

A block in a block chain is said to have n confirmations if it is a part of the valid chain with n blocks. It's generally an assumption that a transaction with enough number of confirmations is safe from double spending.

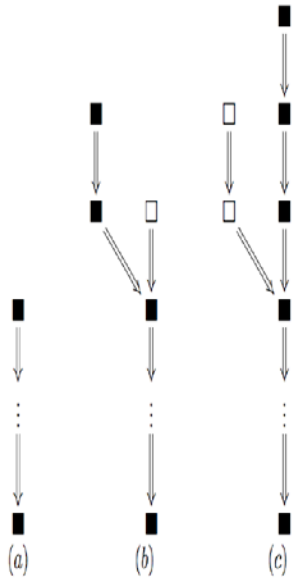


Figure 2: Outline of double-spending. (a) The state of the block chain when the attack starts. The leaf block does not have any of the relevant transactions yet. (b) The branch on the left is known to the network, and includes the transaction paying the merchant with 2 confirmations. The merchant now sends the product. Meanwhile, the attacker has found 1 block in an alternative private branch which credits himself instead. (c) If the attacker manages to get his branch to be longer than the one known by the network, he releases it and the payment to himself is now accepted by the network.

A merchant generally waits for n confirmations of the paying users transaction and then provides the product. The chances of the attacker successfully trying to double spend depends upon his disadvantage i.e. the value of z at the time n confirmations are reached. Here we use a model m more accurately as a negative binomial variable. It is the number of successes (blocks found by the attacker) before n failures success [3]. The probability for a given value of m is

$$P(m) = \binom{m+n-1}{m} p^n q^m.$$

Once n blocks are found by the honest network in a period of time during which m+1 blocks are found by the attacker, the race starts with z=n-m-1.

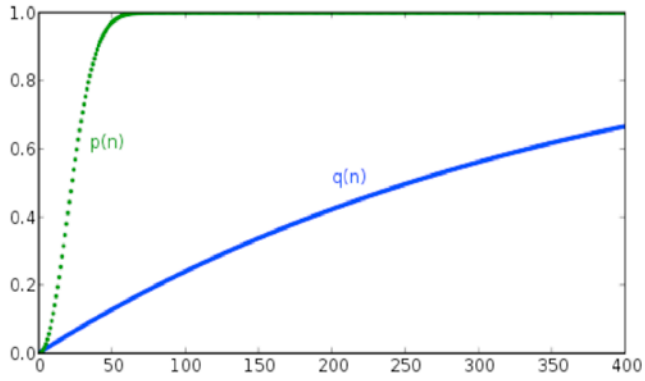
It follows that the probability for the double spend to succeed when the merchant waits for n confirmations is [3]:

$$\begin{aligned} r &= \sum_{m=0}^{\infty} P(m) a_{n-m-1} \\ &= \sum_{m=0}^{n-1} \binom{m+n-1}{m} p^n q^m \left(\min(q/p, 1)\right)^{n-m} + \sum_{m=n}^{\infty} \binom{m+n-1}{m} p^n q^m \\ &= \begin{cases} 1 - \sum_{m=0}^{n} \binom{m+n-1}{m} (p^n q^m - p^m q^n) & \text{if } q < p \\ 1 & \text{if } q \geq p \end{cases} \end{aligned}$$

3. MOMENTUM METHOD

The traditional method of maintaining a hash based proof-of-work as proposed by Satoshi Nakamoto [1] is proved to be effective against majority of the attacking nodes but it has its own disadvantages. In general a proof-of-work schemes are developed in such a way that they are difficult to solve but relatively easier to verify. Most proof-of-work schemes achieve their verification through just one round of an embarrassingly parallel search algorithms [4]. These algorithms quickly adapt themselves to graphic cards, FPGAs, or even ASIC designs. Achieving this would give the attacker a huge advantage over the common computer. In crypto-currencies like Bitcoins, the main motive of the proof-of-work is decentralization of trust. Thus, in such a case it is mandatory that the proof-of-work is able to build resistance towards the acceleration and optimization by various graphic cards, FPGAs or ASIC designs. Hence, to avoid such attacks using custom hardware, we introduce the concept of memory-hard proof-of-work algorithms. One such approach is the use a Sequential Memory Hard Function [5] such as Srypt. But the issue with this approach is that it uses up a lot of memory and as it does so, the ability to be easily verifiable is lost.

To achieve the goal of being trivial to verify but memory oriented to solve, the proof-of-work must have asymmetry in the amount of memory required to validate the work. Algorithms can be made memory-hard by requiring a solution that depends upon the relationship between any two or more parallel steps and thus benefit from the storage of result in every parallel step. Performing just two or three parallel steps and checking results and comparing them can quickly verify the result. The most straightforward example is finding collisions based upon the Birthday Problem [3]. In probability theory, the birthday problem concerns the probability that in a set of n randomly chosen people, some pair of them will have the same birthday. By the pigeonhole principle, the probability reaches 100 per cent when the number of people reaches 367. However, 99 per cent probability is reached with just 57 people, and 50 per cent probability with 23 people. These conclusions assume that each day of the year is equally probable for a birthday.



p(n) = probability of a match q(n) = probability of matching your birthday

Figure 1

The figure above shows us the advantage we get by recording all n solutions while matching birthdays. We observe graphically the benefit we achieve by remembering all n solutions in the search for matching birthdays. If you increase the requirement to finding 3 people with the same birthday then the memory requirements exceed 1 gigabyte on average. Any attempt to replace memory with computation would force the algorithm to follow the q (n) curve, which is at such an algorithmic disadvantage that massive parallelism cannot overcome the need for memory to efficiently solve this issue. The best hope is to generate potential matches in parallel but results will have to be stored for most efficient solutions.

The most straightforward solution would need an array of a few billion items and will have to check the memory after finding a potential result. This approach will require around 500GB of RAM and hence is obviously impractical. Another proposed solution is the use of an hash table which would solve the memory issue but for an attacker, the hash table becomes the source of slow atomic operations. The embarrassingly parallel birthday generation step is restricted by the need to synchronize the storage in the hash table. This last synchronization step places a limit on the amount of parallelism that can be employed [4].

Thus the algorithm being used in this paper basically assumes a cryptographically secure hashing function Hash(x) and a Sequential Memory-hard hashing function Birthday-Hash(x) such as Scrypt. The algorithm for this proof-of-work can be defined as:

Given a block of data D, calculate H = Hash(D). Find nonce A and nonce B such that
BirthdayHash(A +H) == BirthdayHash(B+H)
If Hash(H + A + B) < TargetDifficulty then a result has been found, otherwise keep searching [4].

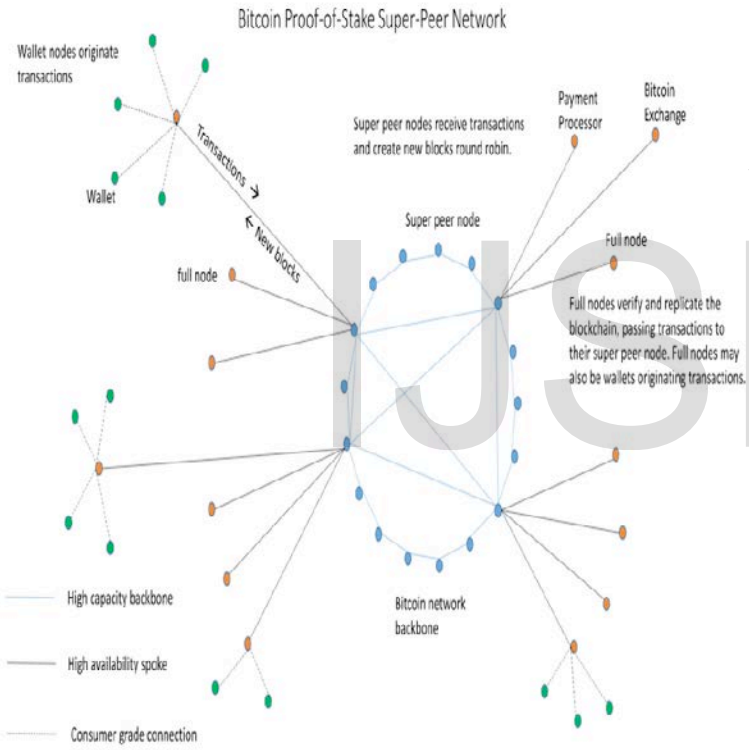
In case of Bitcoins, along with being memory hard, the proof-of-work must also be flexible enough. Hence, due to this, the final step of the proof-of-work is to perform the hash of data and both birthday nonces and check if the result is below a certain value. This final step behaves just like Scrypt based systems. We infer that the best result could be achieved if we combine the birthday search with the traditional memory hard function such as Scrypt.

With traditional proof-of-work systems like SHA256 or Scrypt, it's possible to gain performance by using parallelism alone. However, the use of memory should scale to store the result of every parallel run or the algorithmic disadvantage would be lost. We see that the performance of the proof-of-work increases with the time it runs filling the memory up with potential birthday matches. As a result, this algorithm has some momentum to it making it expensive to restart the search with a new data block. This momentum proves to be very useful for the Bitcoin miners as they could gain some advantage by adjusting the structure of the block being mined every time new data is available. An efficient strategy of mining is to cache all the received transactions while mining the current block until someone finds the current block, and then he could create a new block consisting of all the transactions cached. This property means that a transaction broadcast 5 seconds after the last block was found has no advantage over a transaction broadcast 5 minutes after the last block was found because few miners will begin working on including either transaction until the next block is found. It is because of the momentum property that we have named this proof of work system Momentum[4].

4. PROOF-OF-STAKE METHOD

Satoshi Nakamoto's [1] Bitcoin software assumes a faulty and insecure and adversarial peer-to-peer environment. There is no distinction with respect ot the capabilities of full node peers, nor of the mesh network connections between them. Transactions are relayed and accepted into the block chain on best effort basis. A new transaction reaches 50 per cent of the nodes within 1.2 seconds and 90 per cent within 2.9 seconds [7]. Transactions are not timestamped. There is no definite version of the blockchain. A hard-fork reconfiguration of the peer-to-peer Bitcoin network is described that substitutes tamper-evident logs and proof-of-stake consensus for proof-of-work consensus. Proof-of-stake was first mentioned in July, 2011 by Quantum Mechanic[8] Stake-voting as a method to achieve distributed consensus for building the blockchain was not available as an option to Satoshi because Bitcoin started in 2009 with no stake. In contrast to the original Bitcoin, the system as proposed by Nick Szabo [9] [10][11], exhibits a coordinated behavior. According to him, a trust independent cooperation would be possible through authenticated peer attestation of correct behavior. Consensus is mainly required when misbehavior is detected. The method resists adversaries lacking satisfactory amount of stake. A faulty or misbehaving node is

disconnected from the network by a group of stake-weighted peers. A central mint directly receives the transactions and records its arrivals followed by a broadcast of the accepted transaction to its peers. Since there is single mint, proof-of-work is not needed. As mentioned by Szabo, this mint system is prone to problems such as centralization and lack of redundancy. Failure of the mint causes failure of the entire network. Thus the proof-of-stake system is implemented as a hierarchical network of transparent nomadic software agents executing a common open-source software program with differing roles. There are no humans in the loop except as owners of nodes, which host the agents. The human node owner may offer a controlled bitcoin address-containing stake in return for daily dividends, but is otherwise anonymous [12].



This paid for enterprise class network is configured as a super peer network [13] and is capable of handling transactions all over the world. In this system, certain nodes are given priorities and capabilities superior to the other nodes in the network. These super peer nodes act as the backbone of the network. The traffic flows through these backbone nodes to reach the mint. The accepted transactions and new hashes flow out from the mint to the nodes. Each node has several connections to super peers. Along with the branded entities super peer nodes, full nodes exist that serve hosted wallets, payment processors, exchanges. Such full nodes are likely to be connected to reliable branded super peers.

This system uses an attestable unforgeable log organization inspired by Nick Szabo. This system uses an attested append only memory. It remains correct and keeps making progress even when half the replicas are faulty.

Each full node present in the network has a synchronous replica of the single block chain in the network and consensus agreement as to the latest block hash and the time stamped order of the accepted transactions. All the nodes agree upon the date and time and the calendar of the super peer agent activities. Every full and super node has an identical software capable of performing any particular role. Every node in the network validates and maintains a copy of the single, longest blockchain and the current hash.

A new full node is given a set of peer nodes. This node checks for the fitness of its peers. The new node solicits the members of the set of super-peers from 10 randomly connected nodes. The new node then connects to three super peers, which have low latency provided they are willing to accept additional full nodes. The new node provides justification as to if it should be a member of the super-peer set when asked by the unique configuration node.

The new node provides justification as to if it should be a member of the super-peer set when asked by the unique configuration node [12]. In this algorithm, super peer nodes host a variety of agents. A brief description of these agents is given below:

Configuration Agent: The single configuration agent, which is hosted by a super peer node, has the responsibility of choosing which nodes are super-peers and setting the schedule of the nomadic agents i.e. the agents that haven't been assigned a responsibility as yet.

Seed Agents: The seed agents, hosted by super peer nodes, have the responsibility for seeding full nodes that join the network. The seeded full node selects the seeding agent having the optimal combination of low latency and current load.

Mint Agents: The mint agents hosted by a super peer node has the responsibility of creating new blocks and minting new coins. New blocks are generated at the rate of six blocks per hour. Received transactions are immediately broadcast back into the network so that all full nodes may build the new block in synchronization with the mint agent.

Reward Agent: The reward agent distributes dividends daily to full nodes. By default, full nodes are compensated for their operating costs by a fixed proportion, and additionally receive a reward in proportion to their offered stake.

Audit Agents: The primary audit agent is a singleton having the responsibility of a passive and active auditor. It receives reports of any inconsistencies from any other node, e.g. some disagreement among the nodes, and then investigates it. The secondary audit agents perform random checks that support the functionality of the primary audit agent and in certain cases offloads audit tasks for parallelization.

Recovery Agent: The recovery agent is a single node, which has the responsibility of performing fault-recovery. A typical fault recovery in case of a defective mint would be the rollback of the last committed block, moving its transactions into a replacement block by a temporary mint, which is used as backup.

Software Provisioning Agent: This agent is responsible for coordinating the deployment and reversion of software releases, in particular to super peer nodes.

5. INFERENCES

In the original Bitcoin [1] concept consisting of a hash based proof-of-work chain, attackers (common computers) can double spend the Bitcoins if they construct an alternate block chain faster than the original chain. Given our assumption that p>q (p: probability of an honest node finding the next block, q: probablity the attacker finds the next block) the probability drops exponentially as the number of blocks the attacker has to catch up with increases. Thus, the odds being against him, if the attacker doesn't cover a considerable amount of blocks early on, his chances to catch up become vanishingly small. We thus see what is the probablity the attacker can filtrate the block chain and try to re-spend the money and how we can minimize it. We also suggest much more efficient but complex alternate methods in order to keep the block chain secure. One of the methods is Momentum [4], which is a memory hard proof-of-work which is basically the synchronous combination of the Birthday Problem [6] algorithm, which works towards finding a match, and certain hashing functions, which are also used, in the traditional proof-of-work systems. This method proves to be much more efficient in maintaining a block chain based proof-of-work and protecting it from attacks from external hardware such as graphic cards. Another method, which was suggested, is the Proof-of-Stake [12] method, which has proved to be even more efficient than the method of Momentum. This method basically provides different nodes with different priorities based on the stake they hold. Some nodes are made superior to the other nodes. There is a central mint involved to which data is transferred and which transfers data to the other nodes. The transfer of data to/from the mint is coordinated by a backbone of nodes (certain important nodes which form the backbone). Any nodes in the network lacking stake or misbehaving in the network can be resisted and removed from the network by the other nodes having stake. This proposed system exhibits coordinated and cooperative behavior in contrast to the traditional Bitcoin implementation. The proof-of-stake method is to be put into action by 2016 followed by a year of public testing.

6. CONCLUSION

In this paper we see how a Bitcoin [1] network operates without the interference of a third party/financial institution. It does so by maintaining a publicly available hash based proof-of-work chain (block chain) consisting of all the transaction that have taken place in the network. The longest chain helps in keeping a track of the valid transactions that have taken place in due course of the network thus protecting the network from attacking nodes that look to attack the network by mingling with the transactions and re-spending the money they have already spent (double spending). We protect the chain by making sure the majority of the computational power lies with the honest nodes (Bitcoin Miners) so that the attacking nodes dont take over the control of the majority and infiltrate the network. This method still is susceptible to external hardware such as graphic cards. For this purpose we propose an alternate method known as the Momentum Method [4]. This method is a memory hard proof-of-work system which is a combination of the traditionally used hash functions and the functionality of the Birthday Problem [6] algorithm to give us a new class of proof-of-work algorithms that asymmetric in memory and time for finding a solution in comparisons to verifying the solution and which contains a significant amount of sequential operations. We also found a completely new but complex method to implement the proof-of-work in a much more efficient manner. This is the Proof-of-Stake [12] method in which the network consists of a central mint and the other nodes in the network are given superior priorities. Some nodes are made Super nodes or different agents that provide different functions and the other nodes are the full nodes. A few high priority nodes form the backbone of the network that transfer data to and from the central mint. The division of the nodes is done on the basis of the amount of stake held by each node and the rewards from the proof-of-work are split accordingly. Thus this method as we can see is based on the teamwork of the nodes. This proposed method is to be deployed in 2016 followed by a year of public testing. We see that this Proof-of-Stake method, though complex, proves to be the most effective and offers compelling benefits when it comes to maintaining and protecting the hash based proof-of-work chain in the Bitcoin network.

REFERENCES

[1] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2009.

[2] W. Feller, "An introduction to probability theory and its applications", 1957.

[3] Meni Rosenfeld, Analysis of Hashrate-based double spending, December 13, 2012.

[4] Daniel Larimer, "Momentum- A Memory-Hard Proof-Of-Work via Finding Birthday Collisions, December 2006.

[5] Colin Percival, Stronger Key Derivation Via Sequential Memory-Hard Functions, October 2003.

[6] Roberts Matthews, Fiona Stones, Coincidences: the truth is out there, 2008.

[7] BitcoinStats, http://bitcoinstats.com/network/propagation/

[8] QuantumMechanic, Proof of Stake Instead of Proof of Work, July 10, 2011.

[9] Nick Szabo, Distributing Authorities and Verifying Their Claims, 1997.

[10] Nick Szabo, The God Protocols, 1999.

[11] Nick Szabo, Confidential Auditing, 1998

[12] Stephen L. Reed, Bitcoin Cooperative Proof-of-stake, May 21, 2014.

[13] Beverly Yang, Hector Garcia-Molina, Designing a super-peer network, 2003.