Floating-Point Nakamoto Consensus

Abstract — Public blockchain networks have shown that Nakamoto Consensus is useful in the formation of long-term global agreement — and issues with short-term disagreement which can lead to re-organization (“or-org”) of the blockchain. In this paper we introduce a novel attack against p2p networks relying upon Nakamoto Blockchain Consensus, called a Blockchain Splitting attack (dubbed “ChainSplit”). ChainSplit is the exploitation of byzentine fault injection to conduct double-spend attacks against merchants and exchanges or hold an entire blockchain network for ransom. Adopting a block fitness test in the form of Floating-Point Nakamoto Consensus makes the entire class of Blockchain Splitting attacks much less feasible and reduces the average time needed to achieve a global network consensus over traditional Nakamoto Consensus.

Introduction

Satoshi Nakamoto’s Bitcoin protocol was created to provide a decentralized consensus on a fully distributed p2p network. A problem arises when more than one proof of work is presented as the block in the blockchain as both proofs are seen as authoritative equals. As a result a node will simply adopt the first solution seen, creating a kind of race condition. Byzentiene faults can form when visibility in a distributed and untrusted network is inconsistent. When two segments of the network disagree it creates a moment of weakness in which less than 51% of the network’s computational resources are required to keep the network balanced against itself.

Nakamoto Consensus

Nakamoto Consensus is the process of proving computational resources in order to determine eligibility to participate in the decision making process. If the outcome of an election were based on one node (or one-IP-address-one-vote), then representation could be subverted by anyone able to allocate many IPs. A Nakamoto Consensus is only formed when the prevailing decision has the greatest proof-of-work effort invested in it. In order for a Nakamoto Consensus to operate, the network must ensure that incentives are aligned such that the resources needed to subvert a proof-of-work based consensus outweigh the resources gained through its exploitation.

A minimal network peer-to-peer structure is required to support Nakamoto Conesus. Messages are broadcast on a best-effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone. This design makes no guarantees that the peers connected do not misrepresent the network, and without a central authority or central view - all peers depend on the data provided by neighboring peers.

Security

Nakamoto Consensus holds no guarantees that it is deterministic. In the short term, we can observe that the Nakamoto Consensus is empirically non-deterministic which is evident by re-organizations (re-org) as a method of resolving disagreements within the network. During a reorganization a blockchain network is at its weakest point, and a 51% attack to take the

network becomes unnecessary. An adversary who can eclipse honest hosts on the network can use this as a means of fault injection to disrupt the normal flow of messages on the network which creates disagreement between nodes. When an exchange needs to confirm a transaction, common assumptions such as confirmation length or time sense confirmation may no longer be applicable during a ChainSplit attack - as this disagreement can remain in place for as long as the adversary continues the attack.

With a truly open and decetlerized network any nodes and come and go as they please. An adversary's only restriction on the number of nodes that can be added to the p2p network is the number of IP addresses that can be purchased from Arin or AWS. Over time, dishonest nodes can be introduced to a network which simply delay the transmission of the discovery of new blocks - which in effect will force miners to continue to search for a competing proof-of-work for that block height. When a competing proof-of-work is broadcasted to the network, the adversary will use it's network influence to split knowledge of the proof-of-work as close to ½ as possible. If the network eclipse is perfect then no computational effort is needed, however nothing is stopping the attacker from adding additional computation resources to make sure the system is in balance. Each time one solution is found on one side of the chain - the attacker needs to ensure that another block generated for the other chain - to keep both sides perfectly in balance. As long as two sides of the network are perfectly in disagreement and generating new blocks - the attacker has intentionally created a hard-fork against the will of the network architects.

The eclipse doesn't need to be perfect - however there are ways to improve this deception. By using dishonest nodes to eclipse the network an adversary is conducting a form of byzentine fault injection - which is the introduction of artificial fragmentation between untrusted nodes. When a miner starts to work on a new block - they will broadcast which side of the chain they have chosen - which will inform malicious nodes which side of the network a given miner is participating in. Although blockchain networks as a whole are resistant to DDoS, individual nodes are not, and therefore targeted Distributed-Denial of Service (DDoS) or Flooding can be used to knock off target honest nodes to isolate and re-form logical groups of nodes within the larger network. When a targeted node recovers from a DDoS attack all of its old connections should be broken, so it will search for nodes that are still accepting new connections. An adversary has another opportunity to establish a connection with an honest node when it looks to re-join the network. Using dishonest nodes to flood and eclipse the network an attacker can monitor and reshape mining capacity to construct an eigenvector of computational effort - a balanced byzentine fault created for the purposes of exploitation.

During a blockchain split, each side of the network will honor a separate merkel-tree formation and therefore a separate ledger of transactions. At this point, an adversary will calculate the weaker side of the network. An adversary will then broadcast currency deposits to public exchanges, but only on the weaker side, leaving the stronger side with no transaction from the adversary. Any exchange that confirms one of these deposits is relying upon nodes that have been entirely eclipsed so that they cannot see the competing chain - at this point anyone looking

to confirm a transaction is vulnerable to a double spend. With this ephemeral-currency deposited, the attacker can wire out the account balance on a different blockchain - such as Tether. When the weaker chain collapses, the transaction that the exchange acted upon is no longer codified in blockchain's global ledger, and was replaced with a version of the that did not contain these deposits.

DeFi (Decentralized Finance) and smart contract obligations depend on network stability and determinism. Failure to pay contracts, such as what happened on black thursday resulted in secured loans accidentally falling into redemption. The transactions used by a smart contract are intended to be completed quickly and the outcome is irreversible. However, if the blockchain network has split then a contract may fire and have it’s side-effects execute only to have the transaction on the ledger to be replaced. Another example is that a hard-fork might cause the payer of a smart contract to default - as the transaction that they broadcasted ended up being on the weaker chain that lost. Some smart contracts, such as collateral backed loans have a redemption clause which would force the borrower on the loan to lose their deposit entirely.

With two sides of the network balanced against each other - an attacker has split the blockchain and this split can last for as long as the attacker is able to exert the computational power to ensure that proof-of-work blocks are regularly found on both sides of the network. Although in order to mount this attack it will require the adversary to possess significant computational resources, it is however far less than a 51% attack - thereby defeating the security guarantees needed for a decentralized untrusted payment network to function. When a node finds a longer chain - this hash id is broadcast to neighboring nodes. The nodes conducting this broadcast merely have to be unique IP addresses - they are not required to present a proof-of-work. An adversary with a sufficiently large network of dishonest bots could use this to take a tally of which miners are participating in which side of the network split. This will create an attacker-controlled hard fork of the network with two mutually exclusive merkle trees. Whereby the duration of this split is arbitrary, and the decision in which chain to collapse is up to the individual with the most IP address, not the most computation.

In Satoshi Nakamoto’s view was that the electorate should be represented by computational effort in the form of a proof-of-work, and only these nodes can participate in the consues process. However, SplitChain shows that the electorate can be misled by non-voting nodes which can reshape the network to benefit an individual adversary.

Chain Fitness

Any solution to ChainSplit and byzentine fault injection needs to be fully decentralized. ChainSplit is possible because there is ambiguity in the Nakamoto proof-of-work, which creates the environment for a race conditions to form. To resolve this, Floating-Point Nakamoto Consensus introduces a method of disagreement resolution by setting up a kind of relay-race - where the winning team’s strength is carried forward. This design is intended to cement the

lead of the winner and to greatly incentivize the network to adopt the dominant chain no matter how many valid solutions are advertised, or what order they arrive.

The first step in this new algorithm is that a node in the network should continue to conduct traditional Nakamoto Consensus, but If at any point there are two solution blocks advertised for the same height - then the more-fit block is chosen as the winner which is then propagated to neighbors. It is in the best interest of all miners to adopt the most-fit block, failing to do so risks wasting resources on a mining a block that would be discarded.

In order to have a decentralized solution, this kind of agreement must be empirically derived from the existing proof-of-work so that it is identically verifiable by all nodes on the network. Additionally, this fitness evaluation needs to ensure that no two competing solutions can be numerically equivalent.

Let us suppose that two or more valid solutions will be proposed for the same block. To weigh the value of a given solution, let's consider a solution for block 639254, in which the following hash was proposed:

000000000000000008e33faa94d30cc73aa4fd819e58ce55970e7db82e10f8

There are 19 zeros, and the remaining hash in base 16 starts with 9e3 and ends with f8. This can value can be represented in floating point as:

19.847052573336114130069196154809453027792121882588614904

To simplify further lets give this block a single whole number to represent one complete solution, and use a rounded floating-point value to represent some fraction of additional work exerted by the miner.

1.847

Now let us suppose that a few minutes later another solution is advertised to the network shown in base16 below:

0000000000000000000028285ed9bd2c774136af8e8b90ca1bbb0caa36544fbc2

The solution above also has 19 prefixed zeros, and is being broadcast for the same blockheight value of 639254 - and a fitness score of 1.282. With Nakamoto Consensus both of these solutions would be equivalent and a given node would adopt the one that it received first. In Floating-Post Nakamoto Consensus, we compare the fitness scores and keep the highest.

With both solutions circulating in the network - any node who has received both proof of works should know 1.847 is the current highest value, and shouldn’t need to validate any lower-valued solution. In fact this fitness value has a high degree of confidence that it won’t be unseated by a larger value - being able to produce a proof-of-work with 19 0’s and a decimal above 1.847 is non-trivial. As time passes any nodes that received a proof-of-work with a value 1.204 - their view of the network should erode as these nodes adopt the 1.847 version of the blockchain.

All nodes are incentivized to support the solution with the highest fitness value - irregardless of which order these proof-of-work were validated. Miners are incentivized to support the dominant chain which helps preserve the global consensus.

To explore the extremes of byzantine fault tolerance in the context of an adversary, or to be more specific let's look at the entire class of Blockchain Splitting attacks. In this scenario we must assume a fragmented network where some node have gotten one or both of the solutions. In the case of nodes that received the proof-of-work solution with a fitness of 1.847, they will be happily mining on this version of the blockchain. The nodes that have gotten both 1.847 and 1.240 will still be mining for the 1.847 domainite version, ensuring a dominant chain. However, we must assume some parts of the network never got the message about 1.847 proof of work, and instead continued to mine using a value of 1.240 as the previous block. Now, let's say this group of isolated miners manages to present a new conflicting proof-of-work solution for 639255:

0000000000000000000058d8ebeb076584bb5853c80111bc06b5ada35463091a6

The above base16 block has a fitness score of 1.532 The fitness value for the previous block 639254 is added together:

1.240+1.532 = 2.772

In this specific case, no other solution has been broadcast for block height 639255 - putting the weaker branch in the lead. If the weaker branch is sufficiently lucky, and finds a solution before the dominant branch then this solution will have a higher overall fitness score, and this solution will propagate as it has the higher value. This is also important for transactions on the network as they benefit from using the most recently formed block - which will have the highest local fitness score at the time of its discovery. At this junction, the weaker branch has an opportunity to prevail enterally thus ending the split.

To explore a worst case scenario. Let us assume that both the weaker group and the dominant group have produced competing proof of works for blocks 639254 and 639255. Let's assume that the dominant group that went with the 1.847 fitness score - also produces a solution with a similar fitness value and advertises the following solution to the network:

0000000000000000000455207e375bf1dac0d483a7442239f1ef2c70d050c113
19.414973649464574877549198290879237036867705594421756179
1.847 + 1.415 = 3.262

A total of 3.262 is still dominant over the lesser 2.772 - in order to overcome this - the 2nd winning block needs to make up for all of the losses in the previous block. In this scenario, in order for the weaker chain to supplant the dominant chain it must overcome a -0.49 point deficit.

In traditional Nakamoto Consensus the nodes would see both forks as essentially authoritative equals which creates a divide in mining capacity while two groups of miners search for the next block. In Floating-Point Nakamoto Consensus any nodes receiving both forks, would prefer to mine on the chain with an overall fitness score of +3.262 - making it even harder for the weaker chain to find miners to compete in any future disagreement. Which is intended to erode support for the weaker chain. This kind of comparison requires an empirical method for determining fitness by miners following the same same system of rules will insure a self-fulfilled outcome. After all nodes adopt the dominant chain normal Nakamoto Consuess can resume without having to take into consideration block fitness. Byztenine faults can be resolved more quickly if the network has a mechanism to resolve ambiguity and de-incentivise dissent.

For a given block - each additional solution reduces the keyspace that another solution can fill. If we assume a gaussian random distribution of fitness blocks - then on average ½ of new solutions will be discarded as they are smaller than the incumbent solution. If this new solution is more fit and dethrones the current solution - then the likelihood of this more fit solution being derhoned is smaller by a factor of log2n.

Soft Fork

Blockchain networks that would like to improve the consensus generation method by adding a fitness test should be able to do so using a “Soft Fork” otherwise known as a compatible software update. By contrast a “Hard-Fork” is a separate incompatible network that does not form the same consensus. Both patched, and non-patched nodes can co-exist and non-patched nodes will benefit from a kind of herd immunity in overall network stability. This is because once a small number of nodes start following the same rules then they will become the deciding factor in which chain is chosen. Clients that are using only traditional Nakamoto Consensus will still agree with new clients over the total chain length.

Conclusion

Cryptocurrency networks are intended to be impervious to attacks, and adapting, patching and protecting the network is a constant effort. An organized CoinSplit attack against a cryptocurrency network will undermine the guarantees that blockchain developers are expecting.

Any blockchain using Nakamoto Consensus can be modified to use a fitness constraint such as the one used by a Floating-Point Nakamoto Consensus. This is a general extension of the existing consensus process. A given language’s implementation of using a base16 numeric value vs a floating point datatype is up to the developer - conceptually this fitness test relies upon whole numbers and a partial value, i.e. a floating point value which can be added together.

Nakamoto consensus is only deterministic over time. One could imagine a world where a network became bifurcated by a fork - now if the network is sufficiently unlucky then this bifurcation could never resolve. Theoretically traditional Nakamoto Consensus holds no

guarantees that it is deterministic. Floating-Point Nakamoto consensus allows the network to form a consensus about new chain formation more quickly by avoiding ambiguity in the value of two competing chains.