Æternity Hyperchains

Recycling power of blockchains

v1.0.0

Grzegorz Uriasz Radosław Rowicki Vjacheslav Brichkovskiy Dimitar Ivanov
Ulf Wiger Yanislav Malahov

August 25, 2020

Abstract

Blockchains rely on decentralized consensus between miners. Several ways of achieving it in a safe, provable, and fair manner have emerged. The oldest approach dating back to Satoshi Nakamoto's whitepaper, known as Proof of Work (PoW) implemented in several cryptocurrencies is extremely expensive and leads to ridiculous energy waste. Cheaper algorithms based on Proof of Stake (PoS) promise similar security to PoW, but are usually prone to exploitation. We propose a novel hybrid between PoW and PoS based on BitcoinNG, which combines advantages of both without acquiring their drawbacks.

1 Introduction

One of the most important problems that blockchains need to tackle to maintain a stable and safe infrastructure is the need for a decentralized consensus among the participants. This usually reduces to the existence of a point of agreement, from which everyone should be able to derive an objective and consistent truth about the state of the system. Most commonly, such point defines a certain description of the universe's history, or establishes the person responsible for dictating it for some time. There is a global tendency for depending on some random, fair and unpredictable event that allows for building up a trustless agreement on the current status of events.

Proof of Work (PoW) solutions are traditionally the most popular. They achieve network security by burning (usually empty) CPU cycles in order to randomly hand the power to users depending on their computational effort. This is a slow and costly process, which relies on having a vast and decentralized network of miners. On the other hand, Proof of Stake (PoS) is much more energy–efficient, but its implementation details matter greatly. It could be subject to the nothing–at–stake problem or stake grinding and thus eventually degrade to an inelegant PoW solution.

We propose a hybrid approach called hyperchains: PoS systems, which rely on existing PoW networks for providing security. The security of the PoW network itself is outside of the scope of this document: it is up to the specific hyperchain setup to choose a secure PoW network since it can never be as secure as the PoW chain. It is worth noting that it ought to be possible to change the PoW system used by the hyperchain further down in its lifetime. From now on, we will refer to the PoW chain as a parent chain and the PoS–like system — as a child chain.

2 Existing solutions

In this section, we describe the existing approaches to the problem along with the problems they face and how they attempt to address them.

2.1 Proof of Work

Proof of Work (or shortly PoW) solves the problem of decision making by forcing the users (here called miners) to solve some hard computational puzzle to validate (here, mine) blocks[11]. The point is to make it hard to dominate the network by a single selfish entity. This solution works as long as nobody holds over 50% of the whole computational power, in which case they could just fork the chain at any point and get ahead of the main history line. This is a serious issue since in most protocols the most difficult chain is considered the proper one. Therefore one needs a lot of participants in the network to make it reasonably safe. Moreover, this solution leads to extreme waste of energy and huge costs — according to some measurements, the whole blockchain environment burns enough energy to power the whole Denmark[9].

This idea does not scale well — it is almost impossible to create a public network from scratch that would not eventually be dominated by some malicious entity. A lot of existing serious blockchains suffer this problem[5]. On the other hand, a network becomes extremely secure once it is popular enough.

2.2 Proof of Stake

While PoW distributes the leadership based on computational power, PoS does it based on so–called stake, which in most cases means token supply, sometimes with additional tweaks[8][1]. The idea is to create a leadership voting system which is activated periodically. Each time an election event occurs, the new leader is randomly selected from the stakeholders (called delegates). The chance of winning an election is proportional to the size of one's stake. This approach does not imply any noticeable energetic overhead and therefore is much more friendly to the environment. It also does not require users to have powerful computers to be able to have some involvement in decision making.

However, PoS comes with some serious issues. First of all, there is the infamous "nothing at stake"[14] problem, which exploits the lack of any cost of the actual mining. In this case, there is no downside to staking several branches simultaneously in case of a fork.

Ensuring that the source of entropy is distributed along the chain, makes all of the elections entirely deterministic and predictable leading to a strategy known commonly as "stake grinding," where the dishonest leader tries to rearrange the transactions to influence the result of the upcoming election.

Next issue is the "long–range attack"[15]. In the very beginning, the stake is scattered among a small group of delegates that together have full control over the chain. After some time, they can cooperate and start a concurrent chain diverging from the main one. This could lead to nasty frauds and would destabilize trust over the chain.

On the other hand, there are multiple approaches to deal with these problems. For instance, to prevent nothing–at–stake, the CASPER protocol introduces a "wrong voting penalty," which punishes voters who support conflicting forks[3]. However, this solution is backed by a finality gadget, which is located on another blockchain anyway. NXT deals with long–range attack by forcefully finalizing all blocks that are older than 720 generations[13]. One must note that this doesn't actually solve the problem, but rather defers it. Moreover, it introduces weak subjectivity since one still needs to trust some entity while entering the network for the first time or after a longer downtime. Ouroboros staking system has managed to reach solid security, but at the cost of very high complexity[7].

These are only a few examples. The general point is that PoS comes with many problems, which, as they are being solved, eventually introduce some new ones. This makes most of the pure PoS networks unreliable, and especially not as reliable as mature PoWs.

3 Hyperchains design

The previous approaches had a lot to offer, but considering their weaknesses it is worth thinking of some alternative solutions. PoW seems to work well only with big computational effort being burned and PoS suffers from a huge amount of security holes. The complicated algorithms required for plugging them usually either do not solve the problem at all, moving it further to another layer of abstraction or introducing extreme complexity on the protocol level.

Here we present a hybrid strategy that benefits from the stability of PoW solutions but offers the scalability of PoS systems. A Hyperchain is a special kind of blockchain that sticks to an already existing chain.

They are going to be called child and parent chain, respectively[10].

The parent chain can be almost any blockchain in the world. In general, we want to use some big existing PoW based chains (at the time of writing, preferably Bitcoin or Ethereum) to reuse their burned work to maintain the stability of the child chain. We also propose a PoS–like election system to choose leaders on the hyperchain. In this case, however, we employ a very reliable — and, most importantly, unpredictable — source of randomness — the state of the parent chain. The idea is not very new, though — some research into this topic exists already[2].

Having this machinery, it seems natural to start a new election each time a (key)block has been mined on the parent chain. The next leader shall be chosen depending on the hash of that block and selected with chances proportional to their stake. The selection algorithm is abstract over this document — it is up to the hyperchain to define the details.

We define a group of leadership candidates called ”delegates.” Each delegate needs to express their will of participation in the upcoming election by publishing a commitment transaction onto the parent chain. It is important that they clearly declare their view of the child chain and over which block they are going to compete. Therefore, the commitment must consist of:

- The subject of delegation on the child chain
- The block over which the delegate is going to build
- Signature of the delegate from the child chain

One of the concepts key to the commitment idea is to be able to rely on the parent chain's stability. We want to treat it as a rigid skeleton of the hyperchain, which can be achieved by proper block hash linking. The elected leader will be required to publish the key block on the child chain with a cryptographic proof (referencing the parent) of their right to lead the upcoming generation and publish micro blocks.

One dilemma that rises at this point is whether the commitment should reference the latest key block, or the micro block of the child chain. Referencing a micro block may seem more transparent, but we believe that it would lead to massive forking (especially when some peers would not receive all of the blocks). One must also consider the parent network's throughput. It may be impossible to fit such a huge amount of commitments if we let delegates compete over each micro block. The problem with referencing a key block is that the next leader could steal the transactions and post them in their micro blocks. This, however, can be resolved with a smarter feeing strategy. For instance, instead of giving the full fee to the miner, we can split it up and give the bigger part to the next leader who did include the previous leader's micro blocks in their continuation of the history, as it happens in the BitcoinNG[12]. Following their election, the new leader posts a key block on the child chain, which references the point on the parent, thus proving their right to lead the generation. Besides that, they need to reference the micro block from the previous generation, on which they want to mine.

Figure 1: Hyperchains interface

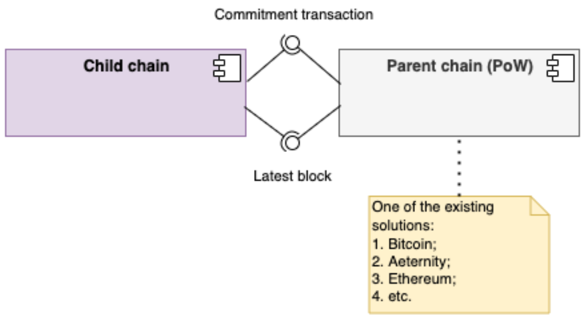
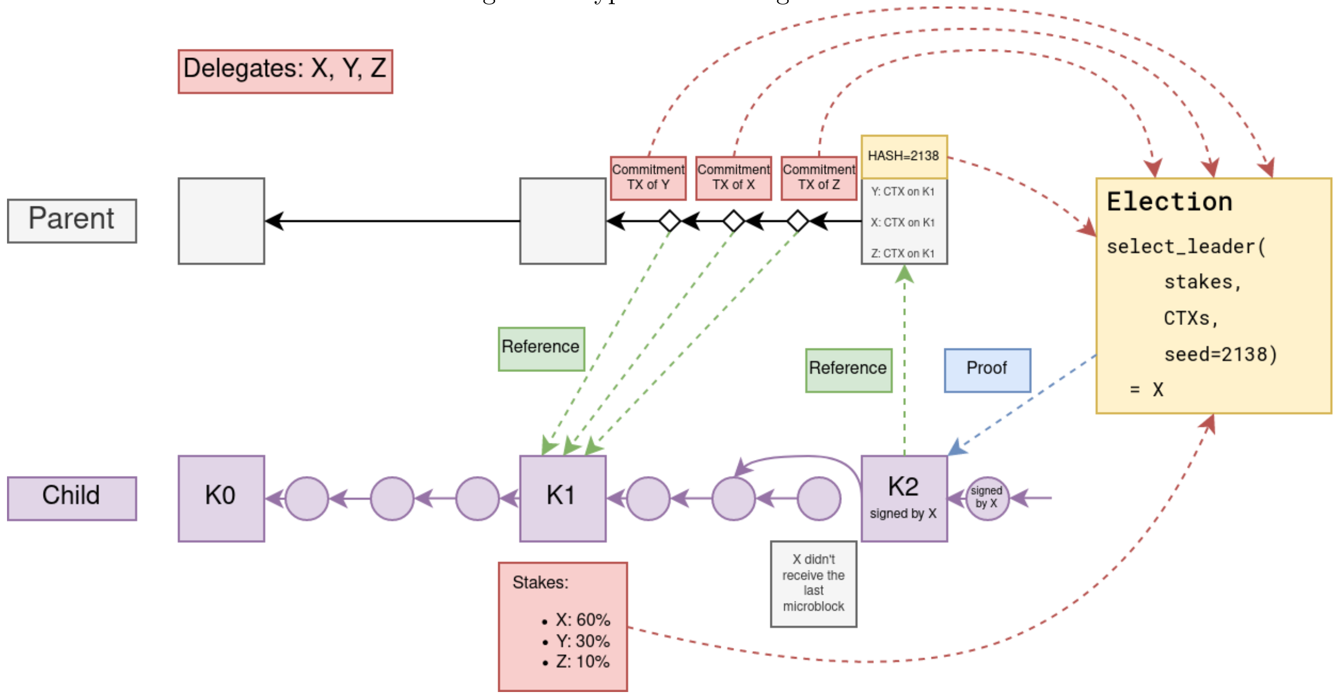


Figure 2: Hyperchains Design



4 Election process and staking mechanism

In this section, rather than prescribe specific mechanisms and rules, we simply propose solutions that would facilitate planning and implementation of the desired algorithms. It is not up this document to specify the details.

The most convenient way to organize the election process is to create a smart contract on the hyperchain, which would manage the stake and evaluate PoF penalties. This contract shall be referenced in the protocol, and its interface should be specified there. It is advisable that the contract be adjusted by regular calls on the fly — this could prevent some protocol–level hard forking. If the VM supports it, the contract could also forcefully alter the blockchain state (e.g. by using some internal Merkle tree framework). We highly recommend introducing consensus changes with a decent delay to ensure that it will not break during a true hard fork.

We propose the following features of a staking contract:

- Leader election
- Voting power calculation
- Delegates calculation
- Voting power delegation
- Applying punishments
- Withdrawing and depositing the stake
- (optional) Controlled hard forking

The system must implement some mechanisms to prevent abuse of the exposed interface and it must be resistant to low responsiveness of the generation leader. Therefore we highly recommend that some necessary calls (like election initiation or reward claiming) be applied automatically in each key block and be protocol–restricted in order to prevent arbitrary calls to them. The contract may be free of any PoF validation and contain only the algorithm of issuing a punishment to the current leader.

5 Security

This hybrid solution allows us to use the parent chain as a reliable protector against most of the attacks that target the PoS systems[4]. In this section, we assume that the parent chain is secured well, and every user has easy and reliable access to it.

5.1 Nothing at stake

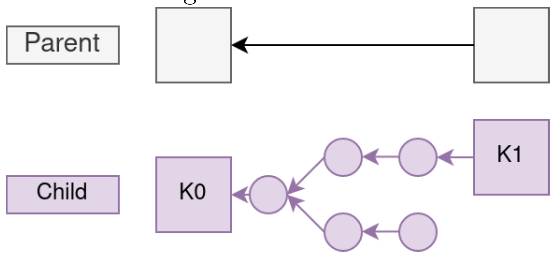
This type of attack splits into two cases: micro forks and generational forks.

5.1.1 Micro forks

A malicious leader may produce blocks without any cost. They are free to create conflicting branches within a single generation.

This case is very similar to BitcoinNG's[6]. We introduce the Proof of Fraud (PoF) mechanism to punish malicious leaders, but in this situation, we can make the penalties much more severe by acting not only on transaction fees but also staked tokens. On the other hand, this kind of forking is not dangerous at all. It may introduce some mess but becomes solved instantly with the next key block.

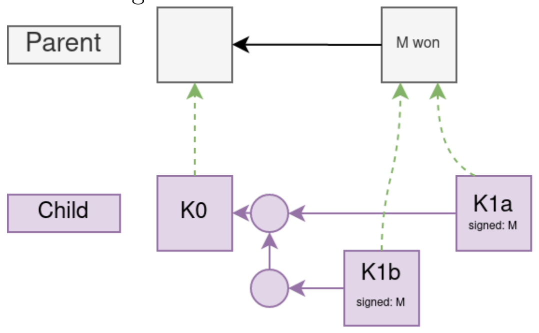
Figure 3: Micro fork



5.1.2 Generational forks

This kind of attack is not a problem in PoW based BitcoinNG as producing key blocks is a hard task. On the contrary key blocks on hyperchains are extremely cheap — a malicious leader may flood the network with coinciding key blocks:

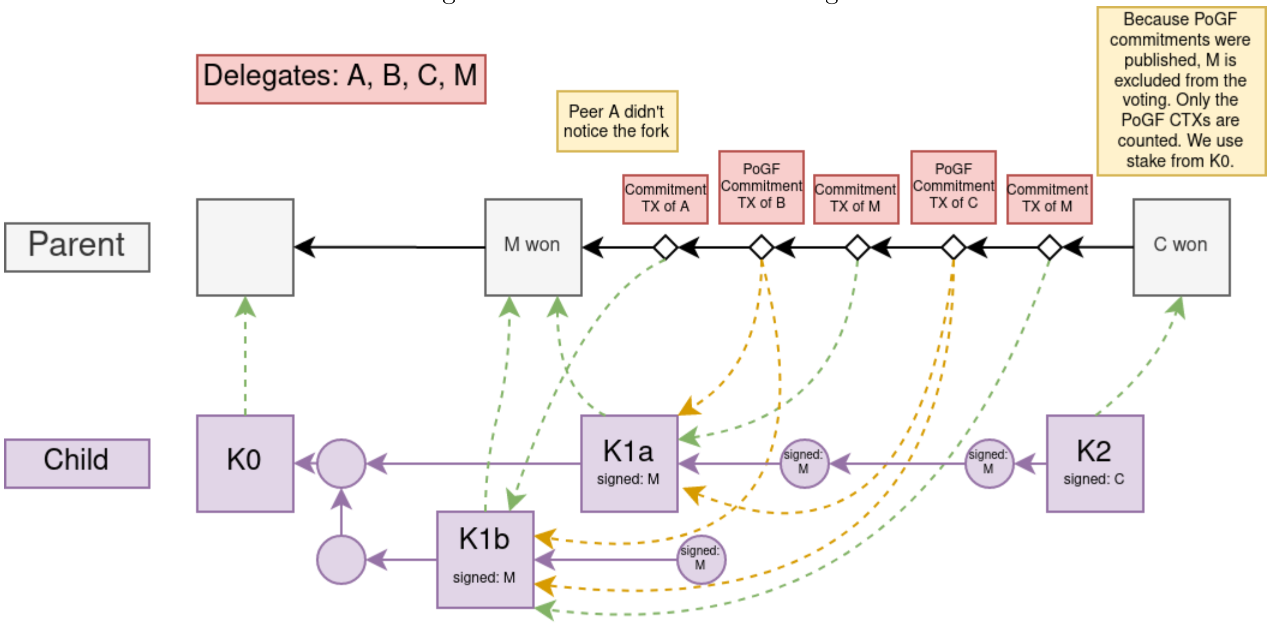
Figure 4: Generational Fork



The $K1a$, $K1b$, etc. are key blocks on the child chain emitted by a malicious leader over different micro blocks. This effectively splits the network into many parts, as it becomes unclear to which of the forks the delegates should commit. To resolve this issue, a single leader must be agreed upon using some additional mechanism — a new election should be performed with the exclusion of the compromised delegate.

To notify the network about the generational fork, the peers need to announce this fact on the parent chain by publishing fraud commitments and cryptographic proofs of generational fraud (PoGF). Each commitment has to point to the latest child key block considered valid by delegates or, in other words, to the generation where the fork starts. The committer also needs to declare to which fork they want to contribute — if they get elected, they will be required to build on it (we disallow rollbacks). The voting power should be calculated based on the latest block before the fraud was detected.

Figure 5: Generational Fork Solving



Due to network propagation delays and connectivity failures, some nodes might not notice that a generational fork was created and not respond accordingly. New nodes or the ones catching up after downtime, need recognize that a generational fork was created and apply the appropriate solution. Some of them may commit to one of the forks and receive a fraud notification after they start mining. Therefore, we introduce a metric over the branches that we call "difficulty" for a conventional reason. The rule to resolve conflicts caused by such data races is similar to the already existing PoW strategy "follow the most difficult chain," and the formula is as follows:

```
difficulty : Block -> Int
difficulty(Genesis) = 0
difficulty(block) =
  if(exists proof of generational fraud on /block/)
        sum of the voting power of delegates
          that committed to /prev(block)/ +
        sum of the voting power of delegates
          that committed to solve the genfork
  else sum of the voting power of delegates
          that committed to /prev(block)/
```

If two forks have the same difficulty, then the one with lower block hash is selected. This formula ensures that key blocks pointing to a PoGF are always strongly preferred over ordinary key blocks — delegates who detect generational forks have higher priority over poorly connected/bootstrapping/syncing ones.

This solution is vulnerable to situations where the leader does not respond or responds with significant delay. In these cases, a new election is held, stalling the network for a while. To resolve the doubts arising when the previous leader reappears, finalization after f (implementation–dependent) generations is introduced.

In particular case a malicious leader may submit a generational fork by publishing key blocks on conflicting micro blocks. In response we prioritize PoGF, because the consequences of forks on key blocks are much more severe than those on micro blocks, and they would need to be resolved anyway.

5.2 Stake grinding

Since the RNG depends ultimately on the key block hash on a PoW chain, it is impossible to predict its outcomes. One could try to mine the parent chain in a special way, but it would require so much computational power that in most cases it would be easier to take control by a 51% attack.

5.3 Long–range attack

While it is still possible to perform a long–range attack, it would be impossible to do it secretly and without preparation since the very beginning. The commitments guarantee that the information of delegates is stored on an immutable chain, and one would need to announce their will of mining suspicious blocks during the entire period of the attack. This would quickly expose the intention of the attacker and let the others prepare for a possible upcoming attack (by blacklisting them, for example).

5.4 Avoiding punishments

Transaction fees may vary depending on circumstances on the network. Consequently, the original penalty system of BitcoinNG is not sufficient in those cases, where the risk of fraud detection is much lesser than expected profit.

One of the most natural ideas is to freeze the stake for some period before the election. In this scenario, the protocol is able to painfully slash malicious leaders by burning/redistributing their stake. However, this may raise some problems when delegated voting is used — malicious leader may vote for their second, empty account that will commit the fraud, losing potentially nothing if compromised. This can be dealt with, allowing only top k stakers to be voted on or slashing *everyone* who supported the malicious leader. We leave this issue implementation–dependent as different solutions require different security approaches.

6 Derived issues

While the presented idea seems to cover the majority of cases, some scenarios may break the system's stability. However, depending on the chosen parent chain and the tweaks of inner protocol, their impact can be limited to an acceptable risk.

6.1 Forks of the parent

Whatever happens to the parent, the similar shall happen to the child. The child chain is entirely vulnerable to forks of the parent chain, and it is quite hard to agree about on which branch to continue. Most likely, the hyperchain would fork as well. On the other hand, if the validators manage to decide on one branch, it would be technically possible to jump into the other if the chosen one becomes less attractive.

6.2 Attacks on the parent

The hyperchain can never be more secure than the parent. If somebody succeeds in a 51% attack on the parent chain, they will also take control of the child chain. Therefore the choice of the parent should be made with regards to its security.

6.3 Finalization time

Since there is no single correct strategy on how to react to forks, the finalization time shall not be shorter on a hyperchain than on the parent chain. If the parent key block gets rolled back, so will all of the leader elections.

7 Conclusion

The proposed solution makes it possible to conveniently create simple, customizable, and most importantly, secure blockchains. The idea of reusing the existing mining power of other networks allows not only small chains to stay safe, but also prevents massive energy waste and, therefore, is much more friendly to the environment. Hyperchains can be utilized in extremely flexible ways. They can be created and maintained almost for free and in case of network overload, they can serve as parent chains for other nested child chains. We leave many implementational details up to the hyperchains creators, because we find them highly use–case dependent.

References

[1] Bentov, I., Gabizon, A., and Mizrahi, A. Cryptocurrencies without proof of work.

[2] Bonneau, J., Clark, J., and Goldfeder, S. On bitcoin as a public randomness source.

[3] Buterin, V., and Griffith, V. Casper the friendly finality gadget.

[4] DEIRMENTZOGLOU, E., PAPAKYRIAKOPOULOS, G., and PATSAKIS, C. A survey on long-range attacks forproof of stake protocols.

[5] Dickman, T. Pow 51% attack cost.

[6] Eyal, I., Gencer, A. E., Sirer, E. G., and van Renesse, R. Bitcoin-ng: A scalable blockchain protocol.

[7] Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol.

[8] King, S., and Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake.

[9] Lee, T. B. Bitcoin's insane energy consumption, explained, 2017.

[10] Malahov, Y. G. Hyperchains — secure, cheap & scalable blockchain technology for everyone, 2016.

[11] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.

[12] Niu, J., Wang, Z., Gai, F., and Feng, C. Incentive analysis of bitcoin-ng, revisited.

[13] Nxt community. Nxt whitepaper, 2016.

[14] Sharma, A. Understanding proof of stake through it's flaws. part 2 — 'nothing's at stake', 2018.

[15] Sharma, A. Understanding proof of stake through it's flaws. part 3 — 'long range attacks', 2018.