onzecurrency crowd: Sybil attackable identities blockchain with poco concensus algorithm

Nico Verrijdt
nicoverrijdt@gmail.com

September 21, 2021

Abstract

We present here another take on a fully decentralised blockchain with a new kind of consensus algorithm called poco or Proof Of Chosen Ones. The goal of this blockchain is to propose a new blockchain primitive and to propose improvements compared to proof of work algorithms (like bitcoin, ...) and proof of stake algorithms (like cardano, ...). In the former case poco has an acceptable energy usage, there's no need for a massive amount of specialised hardware. In the latter case there's a fairer distribution of rewards for the chosen ones, who are chosen ad random out of all the users, when sifting to a final block (comparable to mining/staking). The network is divided between the chosen ones and their purpose is to take care of the communication towards the rest of the network. The chosen ones represent a probability of block acceptance, either they inform the underlying network or they don't. Regarding the proceedings of the reference implementation there will be first a blockchain for identities and in a later stadium a blockchain for simple financial transactions. The proposed blockchain is primarily susceptible to Sybil attacks.

Keywords: poco; blockchain; consensus; identity

1 Introduction

Below we'll explain the poco consensus algorithm.
First we'll explain the assembly of the full_hash, which is the user_id, as a base building block, we follow by explaining the roles of the coordinator and the chosen ones in the p2p network.
Then the procedure of creating preliminary blocks and its distribution over the network,

followed by exploring the sifting algorithm until a final block is found and the distribution of a final block over the network.
What happens when a parallel final block is found and what happens when a headless state is reached?
And lastly a word about attacks and a problem, ending in an explanation of the reference implementation and future additions to this implementation.

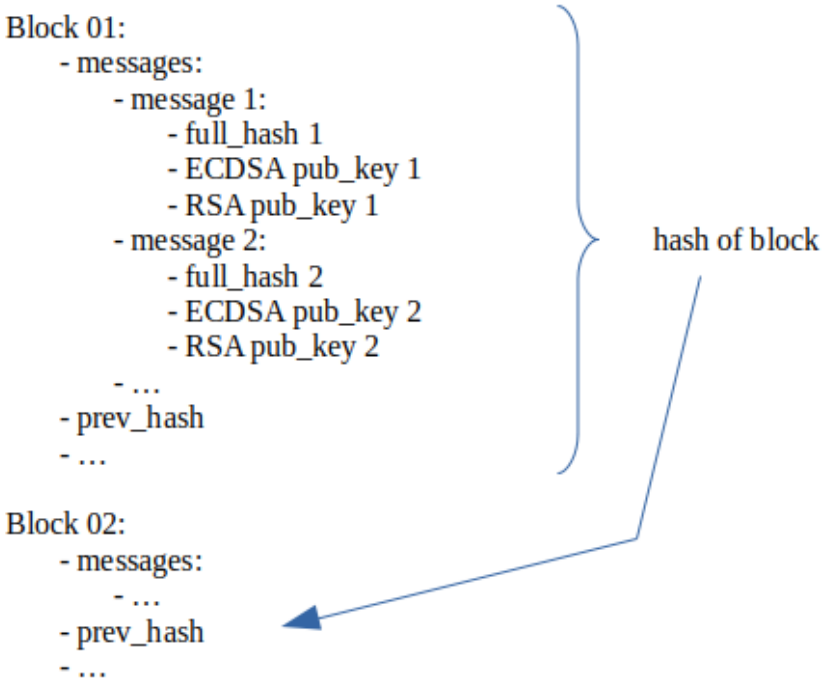
2 Full_hash or user_id

In the blockchain for identities, which is called crowd in the reference implementation, three strings are stored for every user, and these three strings are called a message. First the user_id is stored, which is called full_hash in the reference implementation. Second and third the public keys for ECDSA signatures and RSA encryption are also stored. Keep in mind that this set of three strings is unique for each user and which represents (probably) the weakest point of poco, namely the 51% Sybil attacks, which will be further explained in the attacks section. (remark: for now only ECDSA is used in the reference implementation. Maybe later there will be applications for RSA)

The full_hash or user_id is also unique for each user and is assembled by hashing (SHA256) a concatenated hash of an email address (preliminary) and the hash of the previous block.

$$\mathcal{H}(\mathcal{H}(email) + \mathcal{H}(prev\_block)) = \mathcal{H}(\mathcal{H}(email) + prev\_hash)$$

Figure 1: Contents of a block



Adding the previous hash introduces entropy in the resulting full_hash which might lead to a more equal distribution of full_hash'es in the 2^{256} space of SHA256. It might equally be that this step is not necessary.

Nevertheless, for an even more fair and more equal and even distribution of full_hash'es in the 2^{256} space of the database a counting algorithm can be used (not yet implemented in the reference implementation). This counting algorithm is:

$$shadow\_full\_hash = 2^{256} * \frac{n}{n_{max}}$$

n	place of a user in the full_hash'es database
n_{max}	total amount of users in the full_hash'es database

Figure 2: Sorted full_hash'es in the database

example database:
- entry 1: full_hash = 86
- entry 2: full_hash = 101
- entry 3: full_hash = 506
- ...

The place of a certain full_hash in the shadow_full_hash database might change when

a new user enters the network.

It should be noted that the email address could be replaced by a unique identifier coming from face recognition or fingerprint recognition or a combination. More on this topic in the attacks section.

3 The coordinator and the chosen ones in the p2p network

In the reference implementation the p2p network is divided into 100 buckets (first layer), and every bucket is divided into 100 buckets (second layer) and so on, a bucket consists of maximum 100 users. The users in the first layer of this graph are called the chosen_ones and the first of the chosen_ones is their coordinator.

In section 5 we'll explain how the chosen ones are being used to inform the network.

4 Creating preliminary blocks (comparable to mining)

In a first step for block creation we are going to assemble a lot of preliminary blocks in one block creation period, one of those preliminary blocks will become the final block (discussed in the section about the sifting algorithm). This step is effectively a measure to prevent DDOS attacks.

This might be seen as mining, creating a lot of possible final blocks, but differs from what's commonly seen as mining in the proof of work algorithm.

Those preliminary blocks are created through:

$$prel\_blocks = (\sum_{a=0}^{lbv-1}(\sum_{b=msgs-1}^{0}(\sum_{c=0}^{9}(\sum_{d=0}^{b} msg_d))))$$

of which

$$one\_block = \sum_{d=0}^{b} msg_d$$

lbv	last block vector in block matrix, for calculating the prev_hash (max 100, later 1000 = DDOS prevention)
$msgs$	total messages (max 2048) a messages includes the full_hash, ECDSA pub_key and RSA pub_key
c	counter (10)
d	total incorporated messages, total shrinks to adapt to network invariabilities when a late message isn't received by every user

In practice this formula consists of four for-loops, the first for-loop contains the second, the second contains the third and the third contains the fourth, this is effectively O(n^4) and should be computable because the numbers aren't that big (in round brackets).
In the fourth for-loop the messages are being put in a block with the calculation of a merkle tree root of the incorporated messages, this first loop counts until the total incorporated messages (d) are reached. The third for-loop is a counter from 0 to 9 and used to enlarge the maximum amount of preliminary blocks to enable DDOS prevention. The second for-loop shrinks the total messages (msgs) into total incorporated messages (d). This should improve the probability of truly finding a final block among the greatest share of users, especially when some users didn't receive the latest messages other users might have received due to a personalised delay in the network.
The first for-loop decides the prev_hash used in the succeeding blocks, thus based on the hashes of the blocks in a previous iteration of block creation. This is in fact the chain in the word blockchain.

5 Communicating preliminary blocks

Whenever the network needs to be informed, the procedure is as follows: i.e. a preliminary block needs to be communicated to all (sort of broadcasted). The hash of the block points to a space in the full_hash'es database and the next user to this hash is the coordinator. The coordinator does some controlling and if ok he/she informs the other chosen ones. These chosen ones do some controlling and if all ok the coordinator and chosen ones inform the underlying network layer by layer.
Controlling those blocks means mainly verifying the signature and verifying if all included messages are truthful and comparable to the messages the coordinator or other chosen have received.

6 The sifting algorithm

How this sifting works in practice is pretty straightforward and it's goal is to decide a final block:

- Every user has created all possible preliminary blocks and has saved these blocks, made in one block creation period, in a 2D matrix. After this period a new layer of preliminary blocks are stacked on top of the last layer, the last layer where the hash of a block is the prev_hash in the block of the new layer.
- The communication of preliminary blocks a user sends or receives in the network is registered.
- The 2D block matrix then undergoes a sifting process where the prev_hash in the lastly communicated, sent or received, preliminary blocks must be the hash of the block in the previous layer of the matrix, otherwise if a block's hash of a block is not found in this previous layer then this block is removed (this is the essence of a blockchain). This process continuous through all the layers of the matrix.
- Finally one or more final blocks appear in the matrix almost every block creation period. A final block is a block that remains as single block in that vector of the matrix.

7 Communicating a final block

Communicating a final block to the whole network happens the same as in the case of a preliminary block: the hash of the final block points to a space in the full_hash'es database and the next user to this hash is the coordinator. The coordinator does some controlling and if ok he/she informs the other chosen ones. These chosen ones do some controlling and if all ok the coordinator and chosen ones inform the underlying network layer by layer.
Controlling is the same as for the preliminary blocks.

8 Parallel final blocks and a headless state

When final blocks are found they are communicated in the network. However there's a possibility that parts of the network find other final blocks than another part of the network. This is due to preliminary blocks not being received in the whole network within the block creation period. If other final blocks are found and communicated they will be added to the blockchain as a parallel block.
The user who receives a parallel block will create blocks on top of a all found latest blocks

in fifo order.
This feature is not yet implemented in the reference implementation.

There's also a possibility that a user doesn't receive, nor find, final blocks, for a certain block number, that are correct when verified.
When this happens the blockchain is in a headless state and to solve this a few users should be polled to receive correct information of a correct state of the blockchain and the 2D block matrix.
This is also not yet implemented in the reference implementation.

9 A word about attacks and a problem

- 51% Sybil attacks: The most severe attack possible is the 51% Sybil attacks, which can happen when approximately more than 50% (in reality it's a non-deterministic range) of the users belong to one physical user. He/she has then a greater probability to do malicious things to the blockchain and succeeds in creating a valid final block.
  To keep this problem manageable, new users should be able to use only real physical data like face recognition or fingerprint recognition instead of the here previously proposed email address, but we still have to do some research to know we're able to generate a trustworthy, stable and unique id out of those recognition methods.
  Although the 51% Sybil attacks is a weak point, it might enable cooperation with governments who should have a very good idea of their population and who are able to gather physical data from their citizens. Cooperation with governments is an option, like being regulated is an option, it is for now still an option because onze-currency is truly decentralised and also pseudonimity (a hashed unique recognition string on chain should take care of this) like in bitcoin remains possible.

- 51% compute attack: This attack, as applicable in proof of work blockchains, isn't executable as the longer the preliminary blocks are computed, it is the only compute in poco, the less probable these blocks become final blocks in a normal state of the network.

- Byzantine generals problem: In an ideal world where there's no friction in communication, all the users should receive all the messages immediately.
  In our real world a part of the network might already have received messages while the other part has not. If this happens then there must be a mechanism to shrink the included messages in the generated preliminary blocks, the goal is to find a sweet spot where as much as possible messages are included in the blocks and that the greatest part of the chosen ones for that preliminary block have received all

included messages as this leads to a higher probability of the chosen one accepting this block and thus informing his/hers underlying network.

10 The reference implementation

This paper is written after the reference implementation was coded because we needed to see if it can actually work.
It is written in c++ to create a solid basis for further improvements and there's effectively a lot of space for further improvements.
The code is open source with a MIT licence. We invite you to adapt or build upon the code.
For now, docker is used to setup the software, later on all the different operating systems should be supported as well. Also a graphical environment is lacking, and the implemented terminal interface works just fine.

11 Future additions

- coin, simple financial transactions: The goal is to keep this step very basic in order to be able to understand the double spending problem. Programmable logic won't be included here and is foreseen in the smart contracts step.

- smart contracts: A Turing complete programming language to be able to program the logic you need.

- sharding: To improve the network throughput, sharding will be introduced. If partitioning the users based on their location in a by traceroute generated global graph is doable, otherwise the users will be divided based on the first characters of their full_hash, which is a lot simpler, but the former is what's needed.

- file retrieval system: Something like IPFS for storing the smart contracts in the network, might as well be IPFS itself.

- voting: Enabling global voting is the goal of this whole project. You and us should be able to vote for whatever you find interesting about a happening somewhere. A decentralised application is best suited to implement, manage and direct such a voting.

- governance: The code is what drives this project, the code makes it happen or it just stops. So we need a voting mechanism for the coders and a communication tool for proposing new features and handle bug reports, a github with a voting mechanism as the matter of fact.

Also if one of the coders becomes to powerful or the coders seem to steer the project in a wrong direction, then the public (or the global users) should be able to reset the code to a previous commit.

12 Conclusion

Although there is a chance that Sybil attacks aren't preventable with proof of chosen ones, nonetheless this remains an interesting exercise to broaden the base of possible blockchain solutions.

References

[1] `https://github.com/nvrrdt/onzecurrency` - The reference implementation of onzecurrency.

[2] `https://bitcoin.org/bitcoin.pdf` - A Peer-to-Peer Electronic Cash System - Satoshi Nakamoto

[3] `https://www.peercoin.net/whitepapers/peercoin-paper.pdf` - PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake - Sunny King, Scott Nadal

[4] `https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf` - IPFS - Content Addressed, Versioned, P2P File System - Juan Benet

[5] `https://en.wikipedia.org/wiki/GitHub` and `https://github.com` - GitHub