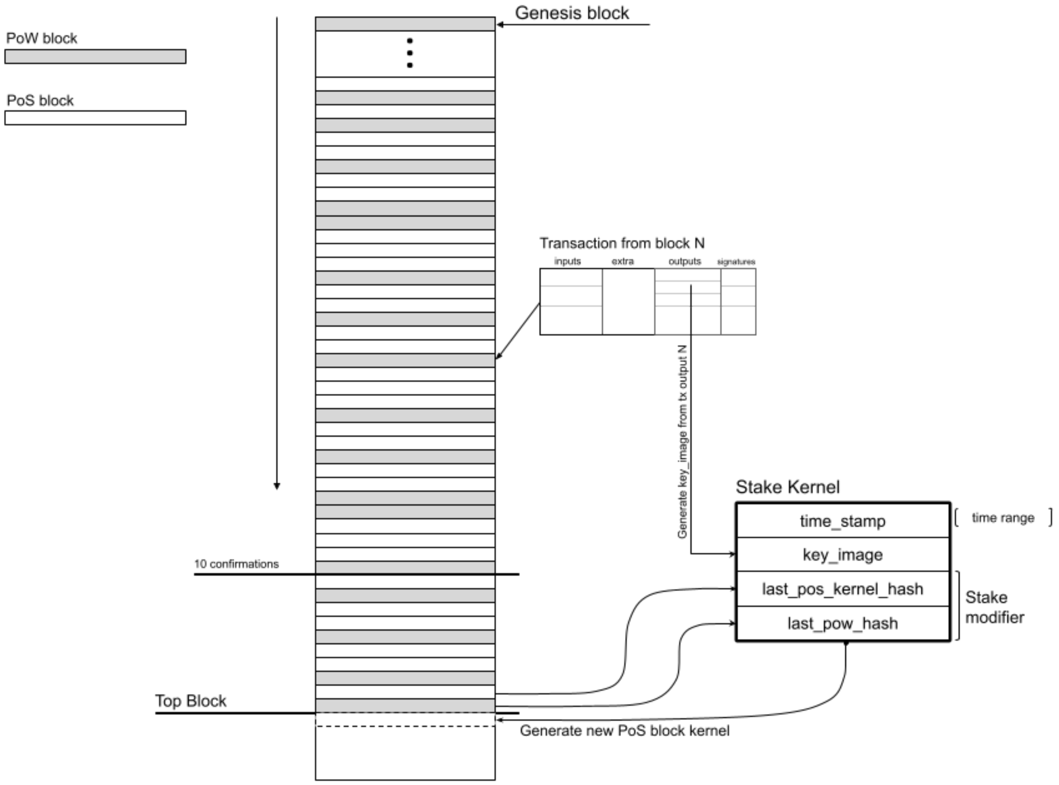
Zano hybrid PoS/PoW improvement proposal

v2.3

Andrey N Sabelnikov (andre@zano.org)
TODO: add acknowledgements before publication
September 2024
www.zano.org

Zano PoS scheme

At Zano, we utilize the Stake Modifier structure, whose hash is checked against the target difficulty of PoS. We rely on the uniqueness of the key_image, which, combined with the timestamp, provides variability in the chances to hit the target and get the opportunity to generate the block. To prevent users from pre-generating PoS blocks by enumeration key images, we also include into the Kernel a structure called **Stake Modifier**, which consists of the last PoW block's hash and the last PoS block's Kernel hash. This structure is visualized in the image below.



Fork choice rule

Zano fork choice rule that was used now based on introducing two balancing coefficients for the PoW and PoS and respectively, which take into account the ratio of cumulative difficulties on two alternative chains. In this case, the total cumulative difficulty can only be calculated relative to another alternative chain, but cannot exist by itself (see Zano whitepaper for more detailed explanation of this formulas).

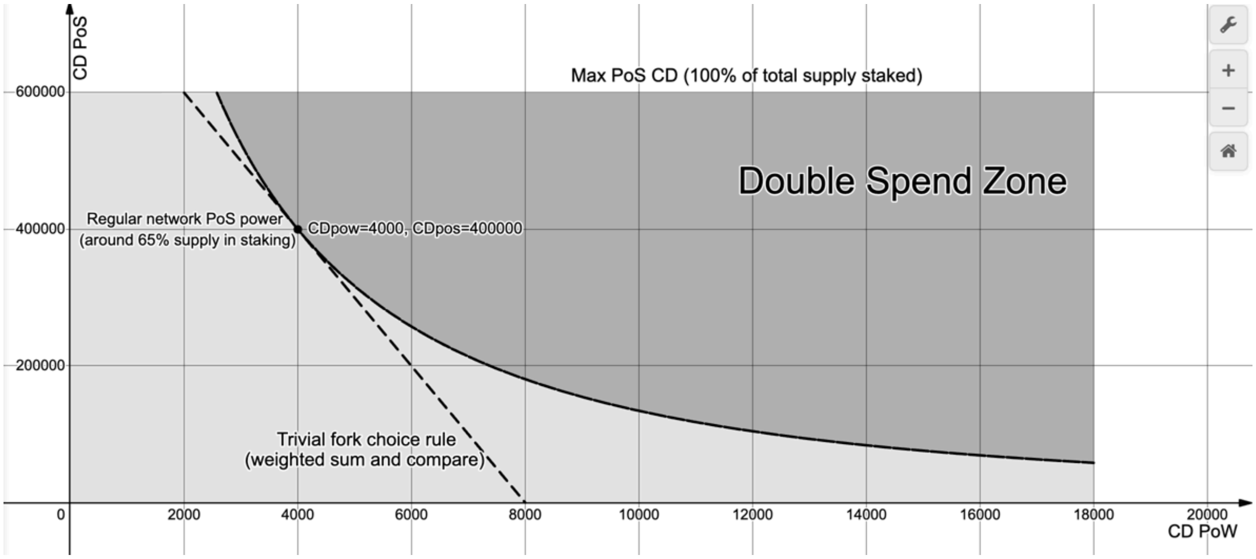
$$K' = \frac{d_i^{pow}}{d_i^{pos}} \; ;$$

$$K_{PoW(X \to Y)} = \frac{CD_{PoW(X)}}{CD_{PoW(Y)}}$$

$$K_{PoS(X \to Y)} = \frac{CD_{PoS(X)}}{CD_{PoS(Y)}}$$

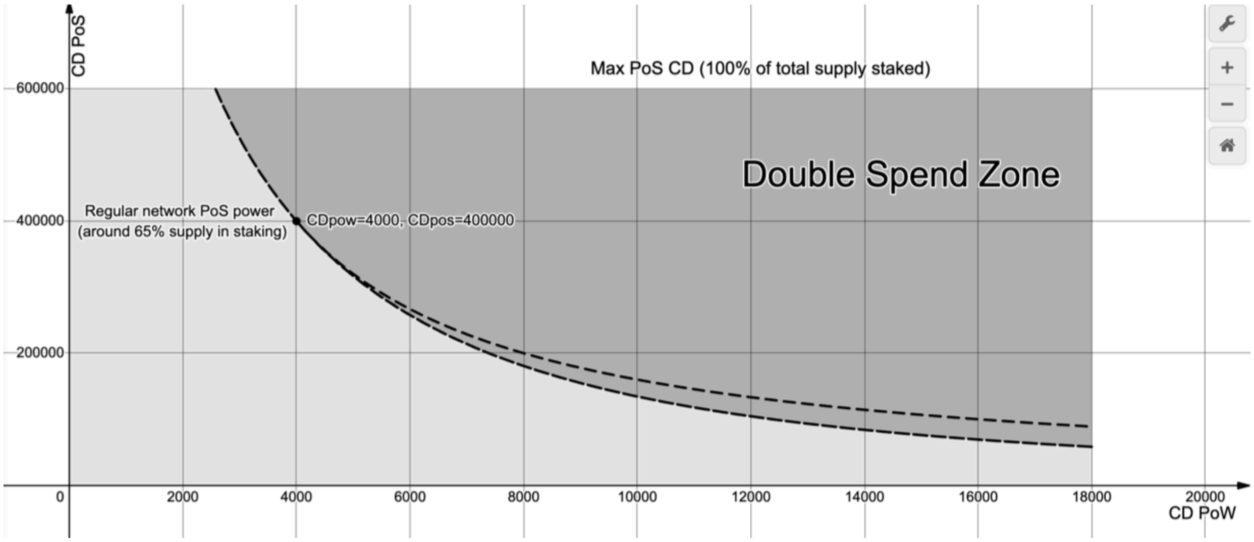
$$\begin{cases} CD_{A \to B} = K_{PoW(A \to B)} \cdot K_{PoS(A \to B)} \cdot \left(CD_{PoW(A)} + CD_{PoS(A)} \cdot K' \right) \\ CD_{B \to A} = K_{PoW(B \to A)} \cdot K_{PoS(B \to A)} \cdot \left(CD_{PoW(B)} + CD_{PoS(B)} \cdot K' \right) \end{cases}$$

The presented fork selection rule forms the following chart, which reflects the balance of resources needed to execute a chain-switching attack:



PoS power multiplication through PoW power problem

Let’s assume PoW miner has the majority of hashrate and can use his/her hashpower to let him/her multiply his chances to find PoS block by not announcing all mined PoW blocks but announcing only those which let him/her win PoS blocks. In this case having 100% hashrate (twice more than needed for performing double spend in pure PoW system) let him do double spend by having only 25% of PoS power, with 200% needed only 12.5% of PoS power and so on. As on Zano fork choice rule that used now this is not an issue since this attack cost line is located above the safety zone. (Dark-grey zone is danger zone)



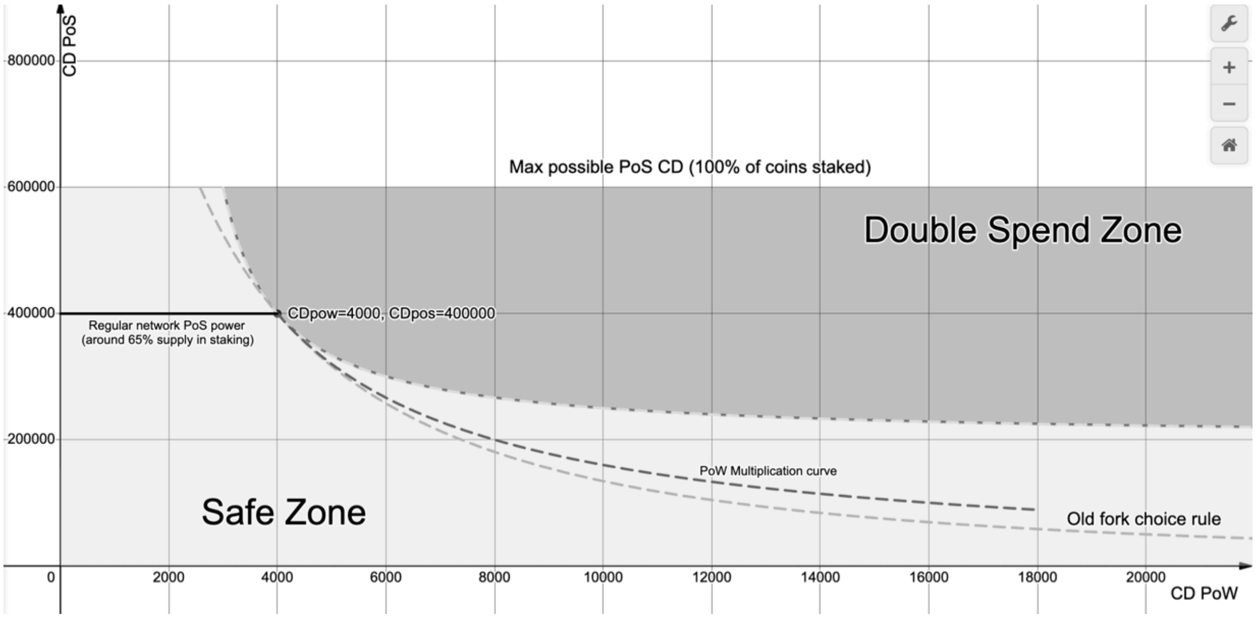
Proposal

Fork choice formula improvement

Assuming that PoS operates correctly, in a hybrid consensus, staking power remains relatively stable, tied to the amount of circulating coins and the percentage of those coins engaged in staking. It is generally accepted that acquiring PoS power externally is impossible, and purchasing a significant share of coins from users becomes economically unviable. Thus, a typical attack against the fork choice rule involves increasing the PoW share while decreasing the PoS share in a competing chain. To defend against this type of attack, we propose an enhanced formula that further resists changes in the PoW/PoS power ratio, relative to the ratio established at the point of the split.

$$\begin{cases} CD_{A\rightarrow B} = K_{PoW(A\rightarrow B)} \cdot K_{PoS(A\rightarrow B)} \cdot \dfrac{1}{(CD_{PoW(A)} + CD_{PoS(A)} \cdot K')^2} \\ \\ CD_{B\rightarrow A} = K_{PoW(B\rightarrow A)} \cdot K_{PoS(B\rightarrow A)} \cdot \dfrac{1}{(CD_{PoW(B)} + CD_{PoS(B)} \cdot K')^2} \end{cases}$$

With this asymptotic formula we're getting more resistant fork choice rule curve[1]:



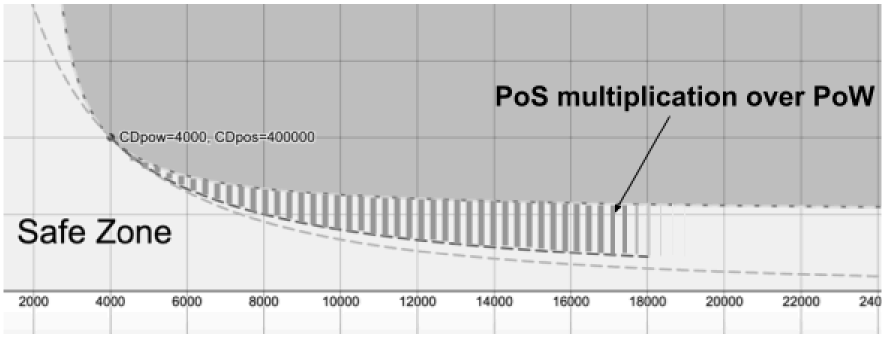
Using this formula and the accompanying graph, it becomes straightforward to theorize about the potential costs of a double-spend attack in terms of cumulative difficulty, which represents the amount of PoS/PoW power applied over time. It is evident that if an attacker can only achieve 60%[2] of the cumulative PoS difficulty (CD), they would need to generate 300% of the PoW CD to compensate. Similarly, with 55% of the PoS CD, they would need 550% of the PoW CD, and at 52%, they would need 1300%. For 51%, the requirement jumps to 2500%, and at 50%, the PoW CD becomes practically unattainable. It's important to note that this analysis is based on cumulative difficulty, which is not the same as hash power or staking power, though it is linearly correlated.

PoS multiplication improvement

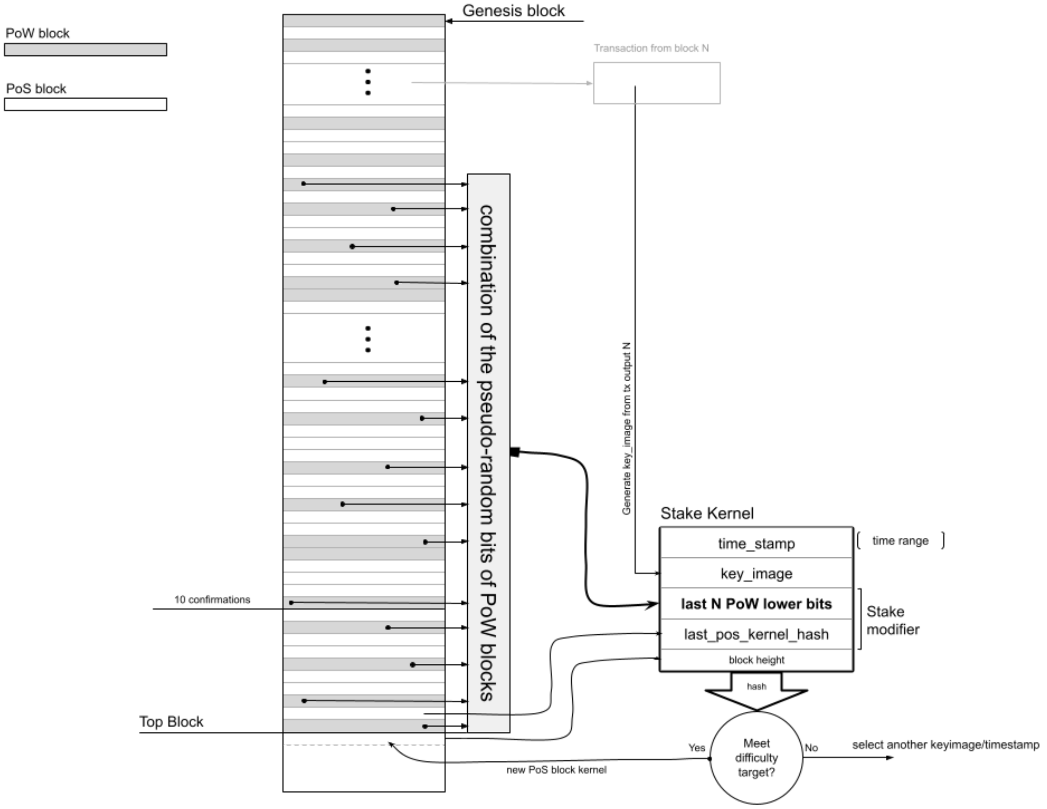
With this improved formula we face another problem: the line that lies between the old and new curves, which indicates the potential zone of a Double Spend Attack through PoS multiplication over PoW power.

[1] Available also at https://www.desmos.com/calculator/6812pqvulv

[2] By 60% means a percent of the cumulative difficulty of the main sub chain from splitting point. Might be confused with "51% attack", which typically refers to a situation when an attacker has hashing/staking power bigger than the rest of the network.



As a solution we propose to replace the PoW component in stake modifier from the last PoW block hash to a set of bits, taken from the last 256 PoW blocks, in a way that each PoW block contributes only 1 bit to the PoW component of the modifier. It might be depicted as follows:



Important details:

1. **Excluding extra PoW blocks.** The bit for the PoW component should only be taken from those PoW blocks that immediately follow a PoS block. Other PoW blocks should not contribute to the stake modifier, in order to prevent the possibility of manipulating the stake through consecutive PoW blocks.
2. **Bits position randomization.** The position of the bit in each PoW block that would be taken for a stake modifier is pseudo-random, and should not be known until the very last

moment, to avoid fixing PoW blocks in favor of some particular key images[3]. For that reason we use last PoS kernel's hash(same as used in new block kernel), and use it to derive position *i* of the bit, that would be taken from block *n*(out of 256 blocks):

$$i = H (H (last\_stake\_modifier) \,||\, n) \bmod 256$$

where
i - is position of bit {0-255} in PoW block *n*,
n - number of PoW block in the array of PoW blocks that are selected to contribute to stake modifier

3. **Timestamps.** The only element intended to be iterated in the kernel is the timestamp. The allowable timestamp range (time window) is recalculated when a new PoS block appears, and it expands over time until another PoS block is found on the network. This is necessary to ensure that PoS blocks are generated over time rather than immediately following a new PoW or PoS block, and also to prevent the network from "stuck" in a situation where none of the key images can generate a kernel that meets the difficulty requirements[4].

Theoretically, a malicious actor could begin generating their own chain of PoW blocks without announcing them, waiting until the timestamp window becomes favorable for their key images, and then attempt to take over the main chain. To mitigate such attacks, it is necessary to limit the timestamp window on the "right side," preventing it from expanding indefinitely.

For instance, if the average PoS window, where the network most frequently finds PoS blocks, is 10 minutes[5], capping this window at a maximum of 100 minutes achieves two things: first, it makes the likelihood of the network getting stuck without finding any PoS blocks very low(TODO estimate probability), and second, it significantly increases the number of coins that would need to be concentrated in control by malicious actor to take over the main chain within a 100-minute window.

4. **Block height in kernel.** If the kernel does not include the block height, then once a staker gains the right to produce a block, their winning kernel remains valid for all subsequent PoW blocks following the last PoS block. Although it is not immediately clear how this could be exploited, it nonetheless raises concerns about the security of the system.

5. **Coin Age and PoS blocks in confirmations.** Even for minimal "coin age" of 10 confirmations[6], it is not clear(yet) how the model described in this work could be attacked. But to ensure security of the consensus, we are considering increasing minimum coin age to some decent level. It's still under consideration, because increasing minimum coin age would cause significant inconvenience for stakers - the wallet's UTXO then should be split to smaller amounts to have optimized staking. Some middle ground coin age might be around 60 confirmations (with 30 PoS

[3] If the position of the bit is known in advance, then the attacker can form any bits sequence he needs for very little cost.

[4] In practice, due to the described architecture, approximately 30-40% of PoS blocks do appear immediately after the preceding block. However, resolving this issue lies outside the scope of this work and is not of interest in terms of the problems being addressed here.

[5] This value is related to the fact that the starting timestamp is slightly in the past and is calculated as the median of the timestamps from the last N blocks to reduce fluctuations and provide initial timestamp variability

[6] At least half of this 10 blocks should be PoS, this is mandatory rule

blocks at least).

References:

1. Zano WP 1.2: https://github.com/hyle-team/docs/blob/master/zano/Zano_WP_latest.pdf