Fair Proof of Stake

A. Begicheva, A. Kofman

May 18, 2018

1 Introduction

From the very beginning up until now Waves has used a "pure" Proof of Stake (PoS) model, as proposed by Nxt [1]. In this model, the choice of account that has the right to generate the next block and receive the corresponding transaction fees is based on the number of tokens in the account. The more tokens that are held in the account, the greater the chance that account will earn the right to generate a block.

In Waves, we are convinced that each participant in the blockchain should participate in the block generation process proportionally his stake. Since we cannot ignore the fact that in Nxt's algorithm this condition is not fulfilled, we have decided to correct the PoS formula. Besides, it can be noted that the current algorithm encourages multi-branching strategies, and this is also a critical issue to be improved. At the moment we do not have the goal of completely changing the algorithm, since there is no need; we simply want to make some adjustments.

In this article, we discuss changes in the forging algorithm of Waves. The aim of the changes to the formula is to correct it so that the process of generating blocks becomes fairer and less vulnerable to multi-branching attacks.

Section 2 describes the current Proof of Stake algorithm and highlights the problems with it. The enhancements in Nxt's PoS model and how to adjust it are proposed in Section 3. Experiments with some Proof of Stake attacks and improvements to the algorithm required to address them are discussed in Section 4.

2 Background

2.1 Proof of Stake (PoS)

Each block on the chain has *generating signature* that depends on the hash of the previous block's generating signature and the public key of the validator's account. Nxt's implementation of PoS is *pseudo-random* i.e. if we know the validator of the previous generated block and the balances of all accounts on the blockchain, we can predict who will generate next block. This is possible

due to a deterministic computation of a block's generating signature, which can be obtained by SHA256 hashing of current block's generating signature and the account's public key. The first 8 bytes of the resulting hash is converted to a number, referred to as the account *hit*.

The base target value aims to adjust the average block generation time to match the desired value (for example, 60 seconds in Nxt). The reason for having the *base target* variable is that not all accounts are online all the time. Sometimes the first account in the queue will not produce a block since it is offline, or its total mining balance is changing, and we need to ensure that the generation time for blocks remains approximately the same. The base target helps to change the complexity of mining for this purpose. The calculation of the base target from [1] is following:

$$(S > 60 \rightarrow T_b = T_p * \frac{min(S, R_{max})}{60}) \wedge$$

$$(S \leq 60 \rightarrow T_b = T_p - T_p * \gamma * \frac{60 - max(S, R_{min})}{60}),$$

where $R_{max} = 67$ - max ratio by which the target is decreased when block time is larger than 60 seconds,$R_{min} = 53$ - min ratio by which the target is increased when block time is smaller than 60 seconds, $\gamma = 0.64$, S is average block time for the last 3 blocks, T_p - previous base target and T_b - calculated base target.

If two or more accounts create a block at the same time, in this case, there is a *collision*, and the less frequently such cases occur, the better it is for the network.

Our current implementation of the Leased Proof-of-Stake (LPoS) Protocol is based on Nxt's mechanism, which can be described with this formula:

$$T_i = \frac{X_n}{(b_i \cdot \Lambda_n)},$$

where T_i is block generation time for i-th account, X_n is a *generating signature* or *hit*, b_i is the proportion of total forging power that the i-th account has, which depends on the account balance, Λ_n is *baseTarget*

For an in-depth analysis of the mathematics and probabilities related to Nxt block forging, see [2], where some disadvantages and bottlenecks of Nxt's PoS system are described. In the article, there is information about such problems as the unfair distribution of the possibility of generating blocks, insufficient resistance to attacks such as Nothing-at-Stake, and the "branching process attack".

The first disadvantage of the current version of PoS is an unfair probability of block generation. An example of such dishonest distribution is shown on the bar chart from Figure 1: the "big guy" with the most coins can create blocks 50% more often than he should, reducing the ability of smaller participants to mine. As we can see in Figure 2 the big guy also gets most of the fees and this is also unfair.

The total rewards received as a result of block generation is the sum of the transaction fees located within the block. The current PoS implementation has

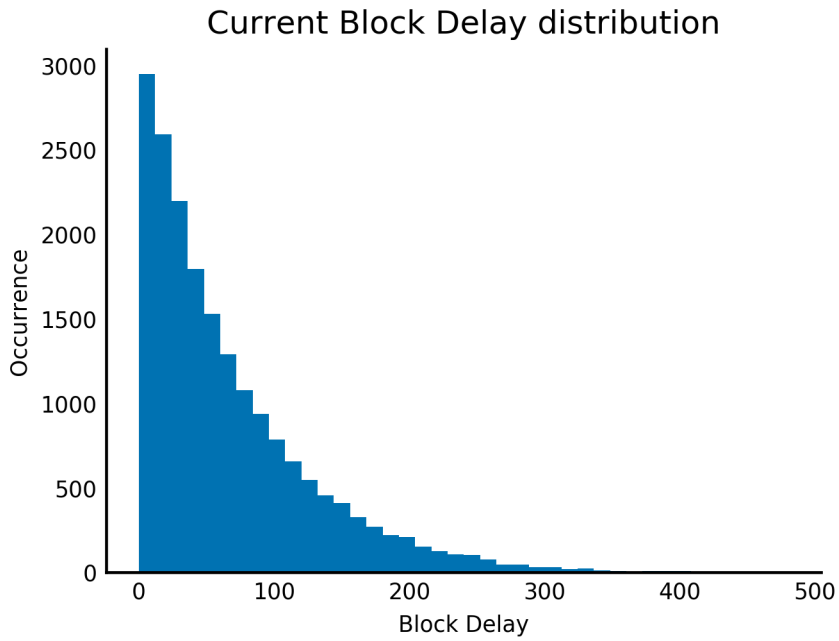

Figure 1: Current block delay distribution in seconds

a distribution of fees between participants, which has an even greater imbalance between large and small participants (3). This is because the average time of the previous block for a large miner is greater than for a small one.

As can be seen from the chart, fee distribution has a drastic drop-off after the biggest miner and the imbalance between a large balance's participants and small participants increases.

2.2 Attacks on Proof of Stake

The main vector for PoS being attacked is the fact that generating a block is no more than generating one signature. The validators have the incentive to work on multiple forks since there is no restriction to do this. Now we need to define what it means for one chain to be "better" than another.

For example, in the Nothing-at-Stake" attack [8] validators could generate conflicting blocks on possible forks with nothing at stake for double-spending and reducing the efficiency of the system. Since the eligibility proof is deterministic for each account, one can easily predict which block will be generated next. A detailed analysis of a multibranch strategy on the basic Nxt algorithm is presented in [3, 5, 6, 4]. These articles show that the multibranch strategy is more efficient in terms of the number of generated blocks and the rewarded fees. If all accounts are indifferent and do not try to detect the attack, the attacker

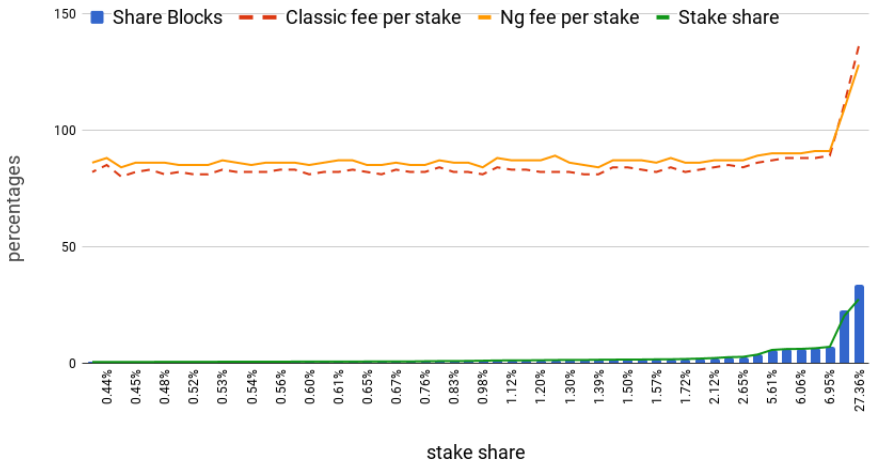

Figure 2: Current fee distribution per balances

always wins with a small forging balance.

A similar attack is Stake-Bleeding [7]. Attackers launch a long-range attack by creating a local copy of the current blockchain along with maintaining an alternative blockchain that is initially empty and is hidden from honest participants. The attackers that produce the blocks receive the fees as a reward, and a large number of the transaction fees in the private blockchain will be collected by the malicious coalition. If the blockchain system has run for a substantial period of time, the transaction fees will turn the attacking minority coalition into a majority that will be able to advance the private blockchain at a speed faster than the honestly-maintained public blockchain. Thus the attacking coalition could rewrite the history of transactions.

In [9] Ethereum presents Slasher, an algorithm which solves this problem. Slasher has harshly punitive nature, and its key feature is that the signing privilege is based on the block mined two thousand blocks ago. Thus, in the event of a fork, a miner that gets lucky in one chain will also get lucky in the other, completely eliminating the probabilistic dual-mining attack. The penalty of block reward loss ensures that every node will take care to sign only one block at each block number.

In [10] two PoS protocols are proposed that are secure against both the Nothing-at-Stake attack and the long-range attack. The idea of both protocols is to restrict the validators to generating at most one block at a given block height. The first protocol is a software-based solution with an enhanced signature scheme which binds the randomness of the signature to the block height value. The second protocol is a hardware-based solution which relies on tamper-resistant hardware and a trusted application for block generation, which records information about the last issued blocks and prevents the validator from repeat-

edly generating an already-generated block.

2.3 Requirements for Improvement

Based on the shortcomings and vulnerabilities of the current implementation of the protocol described above, we propose the following criteria for successfully changing the forging formula:

1. All the chain's participants have a "fair" probability of creating a block, i.e. probability proportional to their stake.
2. The number of collisions decreases, or at least does not increase.
3. Average block generation time is one minute and there are no blocks for which generation time is much greater than this average.
4. Exposure to specific Proof of Stake attacks, e.g. multibranch forging, is decreased.

3 Adjustment of PoS Formula

3.1 Fairness

As shown in [2] unfairness is an essential consequence of simple Nxt formula, which is based Uniform distribution of random value. Authors of this paper also compared the existing formula with a formula based on Exponential distribution, which gives fair results: the probability of forging a block is proportional to miner's balance.

To fix our protocol, we correct the formula and use Exponential distribution instead of Uniform. To do this we apply a logarithmic function to a random value we use:

$$T_i = C \cdot \frac{-\log \frac{X_n}{X_{max}}}{(b_i \cdot \Lambda_n)}.$$

Although average block time is one minute, delays between particular blocks may vary from a few seconds to several minutes. Figure 4 shows block distribution modeled with the new formula. As we can see, there are a lot of blocks with a delay of less than 10 seconds and some blocks appear after more than 600 seconds, or even more.

We would like to avoid having blocks with too high a delay, and ideally to have all blocks appear in not more than 3-5 minutes. Also, it does not make much sense to have blocks appearing a few seconds after the preceding block, since our synchronization protocol (Waves NG) may lead to empty blocks when the block delay is less than 5 seconds.

To fix distribution we apply one more logarithm function, specifically $Log(1+x)$, to the whole block delay formula. It changes only block delay but does not

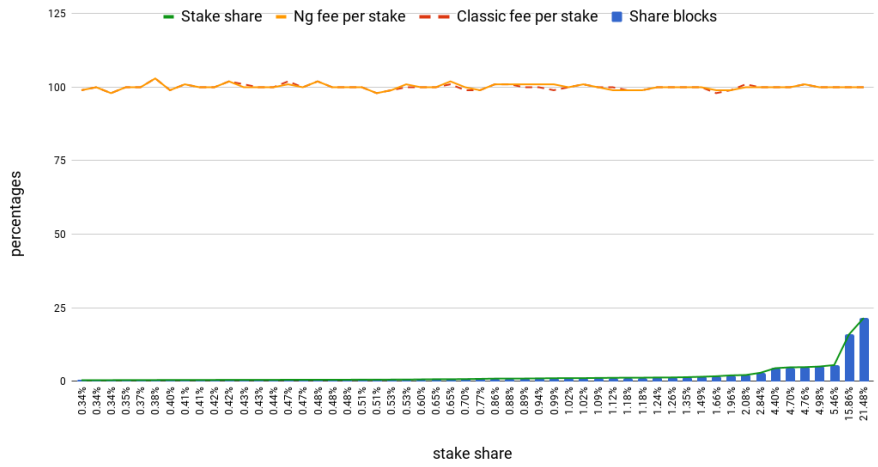

Figure 3: Fair fee distribution per balances

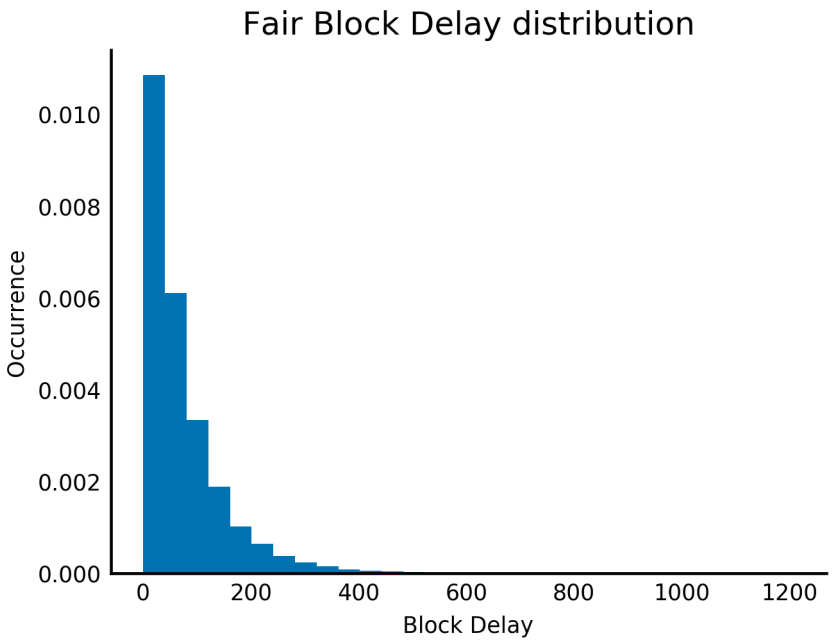

Figure 4: Initial block delay distribution in seconds

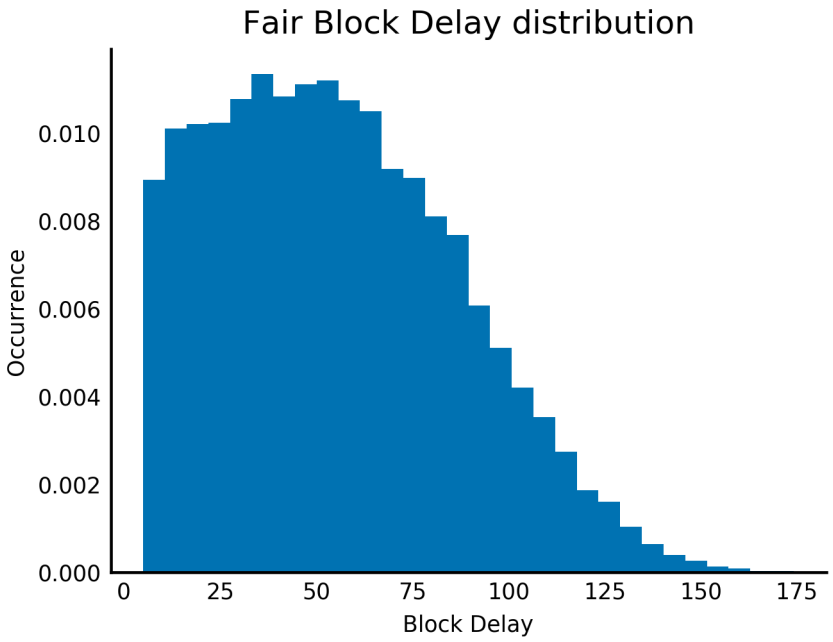

Figure 5: Example of balanced block delay distribution in seconds

affect miner selection, so the fairness property stays the same. Also, we can add some constant, e.g. 5 seconds, to the resulting time to avoid overly frequent blocks.

$$T_i = T_{min} + C_1 \cdot \log(1 - C_2 \cdot \frac{\log \frac{X_n}{X_{max}}}{(b_i \cdot \Lambda_n)}),$$

where $T_{min} = 5$ is a constant for delay between blocks, $C_1 = 70$ - a constant defining shape of delay distribution, and $C_2 = 5E17$ is a constant to adjust base target.

The shape of block delay distribution is now defined by C_1 value. If it's too low we'll have almost all block delays of around 60 seconds, which leads to a high number of collisions (two miners forging blocks at the same time). If it is too high, we'll have a lot of block delays close to T_{min} and some blocks with huge delays. In Figure 5 you can see an example of delay distribution with a balanced value of C_1.

3.2 Adjustment of Base Target calculation

In addition to the basic formula of the PoS, we also change the formula for calculating the base target. We suggest the following formula:

$$(S > R_{max} \rightarrow T_b = T_p + max(1, \frac{T_p}{100}))\wedge$$

$$(S < R_{min}\&\&T_b > 1 \rightarrow T_b = T_p - max(1, \frac{T_p}{100}),$$

where $R_{max} = 90$ - max ratio by which the target is decreased when block time is larger than 60 seconds,$R_{min} = 30$ - min ratio by which the target is increased when block time is smaller than 60 seconds, S is average block time for the last 3 blocks, T_p - previous base target and T_b - calculated base target.

We conducted an experiment aimed at monitoring the dynamics of the base target value over 100000 blocks. A line graphs from Figure 6 and Figure 7 show the base target dynamics of the last 1000 experimental blocks for the old and new formulas, respectively. We changed the network balance for the last 500 blocks, reducing it by half. It can be seen that according to the new formula, a base target value changes more inertly with changes in the network balance. The line corresponding to the behavior of the base target value obtained by the current formula fluctuates quite sharply on a fixed balance, for the first 500 blocks. The formula proposed by us, on the other hand, is resistant to the influence of arbitrary factors and is adjusted gradually to changes. Also, the graphs show an average delay of every 100 blocks for current and adjusted formula. It is noticeable that the current formula instantly smoothes out the block delay when changing the balance, allowing only a slight increase. By contrast, the adjusted formula initiates a significant climb, that is normalized gradually.

The changes in base target formula can be beneficial in a case when a group of miners with a small balance was forked from the main chain. With the new formula, in this case, the fork will lag behind the main chain not only in the score but also in the height.

4 Security Improvement

As our modeling shows, the initial Nxt algorithm is exposed to the Nothing-at-Stake attacks and some combinations of selfish mining and multibranch mining with several accounts. The goal of our further improvement is to decrease exposure to these kinds of attacks by making them considerably less effective. It's important that we improve our current PoS algorithm, rather than switch to different, more complicated one.

In article [2], there is proof that the current algorithm with Uniform distribution is highly susceptible to attack when the attacker has at least 1/3 of all active balances in the network, but that probability can be decreased by using Exponential distribution to replace the old version. With exponential distribution, the attacker must have at least 1/2 of all active balances.

Let us consider following attack: the attacker controls a stake distributed over N forging accounts. Their goal is to make alternative forks win over the main one by overtaking it. Assume that the attacker creates a block after z'_x

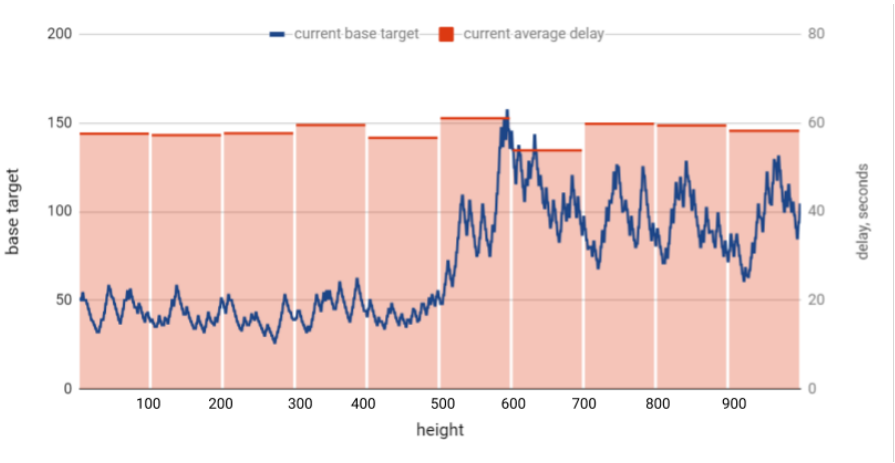

Figure 6: Base target values and average delays using the current formulas.

Figure 7: Base target values and average delays using the adjusted formulas.

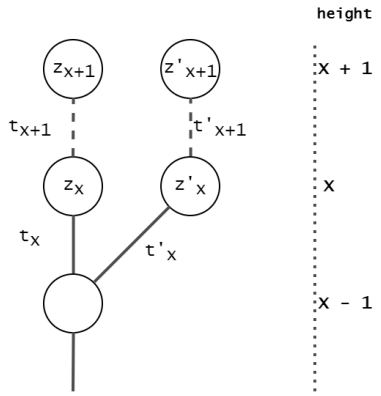

Figure 8

with delay t'_{x+1} and someone create a block on z_x with delay t_{x+1} (8). The fork can become a best chain only in a case when $t'_{x+1} + t'_x < t_{x+1} + t_x$, but we know that $t'_x > t_x$. Thus, to fork became the main chain, t'_x must be such as $t'_{x+1} + t'_x - t_x < t_{x+1}$.

Every time an honest miner forges a new block based on the current best block (parent-block), the attacker generates N hidden chains based on the same parent block. These chains are not best, so they would not be accepted by the network. At the next height, the attacker can extend these N chains with N new blocks each. This way it would have N^2 alternative chains consisting of two blocks. In each chain the delay before the second block is calculated based on a pseudo-random variable extracted from the first block and the forging account of the second block. This means that in order to successfully overtake the main chain an attacker needs to find a better chain out of N^2 chains based on independent random variables. Although it has a lower mining balance the chances of overtaking are rather high due to the large number of alternative chains. The same applies to chains consisting of three blocks, where the number of alternative chains is N^3.

In order to reduce the effectiveness of such an attack we propose not to use a pseudo-random variable from the preceding block, but to take it from an older block. Let us say that the delay for the block at height (h) will be taken from block $(h - 100)$, but not from $(h - 1)$-th block. This will eliminate the combinatorial effect. All the miner's blocks at the same height have the same delay no matter how many chains he tries to extend. Also, the probability of overtaking the main chain by a fork decreases with the increasing height of the main chain, since the attacker has a smaller balance.

Also, we conducted an experiment with this model. We take two miners as attackers (the third and forth from our balances' set), who want to create a fork for various reasons, and run the simulation dozens of times. We launched the old PoS and new PoS models using all the same balances. The results of one of

the simulations are presented in the Table 1, other simulations gave results that did not differ substantially. In the results obtained by us, the attackers received on average 70% more commission in the old PoS model and received only 30% more fees in our proposed model. Also, the Table 2 also presents an example of results that shows the number of forks using data from the Table 1: the number of forks and their length is reduced with the adjusted algorithm.

Table 1: Experimental data: 3-th and 4-th miners are attackers.

			Current PoS			New PoS		
№	Balance	Share	Blocks	Block share	Fee	Blocks	Block share	Fee
1	180	38	35007	35	89	37142	37	96
2	76	16	14740	15	89	15851	16	96
3	41	9	12714	13	155	10255	10	122
4	35	7	11015	11	158	8797	9	123
5	23	5	4544	5	91	4661	5	95
6	18	4	3499	3	89	3654	4	94
7	15	3	2907	3	89	3082	3	95
8	10	2	1893	2	87	2064	2	95
9	7	1	1399	1	92	1449	1	96
10	6	1	1144	1	86	1249	1	96
11	6	1	1138	1	86	1172	1	90
12	5	1	998	1	91	1047	1	96
13	5	1	997	1	91	1070	1	100
14	5	1	986	1	91	1075	1	100
15	4	1	774	1	88	875	1	102
16	4	1	763	1	88	845	1	98
17	4	1	807	1	93	827	1	96
18	4	1	761	1	87	841	1	97
19	3	1	607	1	90	590	1	91
20	3	1	607	1	95	637	1	97
21	3	1	562	1	88	597	1	89
22	3	1	629	1	92	576	1	87
23	3	1	538	1	83	622	1	95
24	2	0	370	0	82	399	0	89
25	2	0	391	0	91	431	0	99
26	1	0	211	0	95	193	0	93

Table 2: Compare the number of forks for the current and adjusted algorithms.

Fork Length	Current PoS count	New PoS count
2	2508	2153
3	794	498
4	326	166
5	142	66
6	71	21
7	31	11
8	9	1
9	8	0
10	7	0
11	2	0
12	1	0
13	1	0
14	2	0
total	3902	2916

Therefore, this enhancement helps us reduce the probability of the attack described. As for the selfish mining attack, where one forger does not publish and distribute a valid solution to the rest of the network and continues to forge the next block and so on, maintaining its lead, our enhancement allows the network to resist the forger, thanks to the fair distribution of forging probability.

5 Conclusion

In this paper, we briefly described the shortcomings of the current PoS algorithm used by Waves that we have decided to address: the unfairness and vulnerability to specific attacks. At the moment, miners who have the largest share of stake create more blocks than they are supposed to, and as a consequence gain much more fees. Besides this, the current PoS implementation does not resist multi-branching attacks well, such as Nothing-At-Stake.

We presented an improved PoS We algorithm that makes the choice of block creator fair and reduces vulnerability to the described attacks, in accordance with the shortcomings of the current algorithm. We analyzed the model of the new algorithm for its correspondence to the stake share and the share of blocks, and the results were positive. Also, the algorithm was analyzed for vulnerability to attacks, and results obtained with the new model were better than with the old one. The attacks' results for the attacker were not so successful in terms of the profits gained. The number of forks and their length decreased.

For future research, there are many other problems of PoS algorithm implementation, for example, that the forging algorithm has a pseudo-random choice of the next block's creator.

References

[1] *Whitepaper:Nxt.* https://nxtwiki.org/wiki/Whitepaper:Nxt

[2] MTHCL *The math of Nxt forging.* (2014) www.docdroid.net/ecmz/forging0-5-2.pdf.html

[3] Andruiman, *PoS forging algorithms: multi-strategy forging and related security issues.* URL https://scribd.com/doc/256072839/PoS-forging-algorithms-multi-strategy-forging-and-related-security-issues

[4] Andruiman, *Nxt forging algorithm: simulating approach.* (2014) https://scribd.com/doc/243341106/Nxt-forging-algorithm-simulating-approach

[5] Andruiman, *PoS forging algorithms: formal approach and multibranch forging.* (2014) https://scribd.com/doc/248208963/Multibranch-forging

[6] Andruiman, *Multibranch forging algorithms: tails-switching effect and chain measures.* (2015) https://scribd.com/doc/256073121/Multibranch-forging-algorithms-tails-switching-effect-and-chain-measures

[7] Gazi P., Kiayias A., Russell A. *Stake-Bleeding Attacks on Proof-of-Stake Blockchains.* https://allquantor.at/blockchainbib/pdf/gazi2018stake.pdf

[8] *Ethereum:Wiki* https://github.com/ethereum/wiki/wiki/Problems

[9] *Slasher: A Punitive Proof-of-Stake Algorithm* https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/

[10] Li, Wenting, et al. *Securing Proof-of-Stake Blockchain Protocols.* Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, Cham, 2017. 297-315.