Cutcoin: a Privacy-Focused Cryptocurrency Based on PoS Consensus Algorithm

Cutcoin Team

July 15, 2019

Abstract

Cryptocurrencies are known for years and have been successfully used in payments around the world. The permanently growing demand leads to the evolution of technologies and new coins come out with advanced functionality. In this paper we present Cutcoin: the first implemented private cryptocurrency with Proof of Stake (PoS) consensus algorithm. We describe the motivation for development, major features and specific details of realization.

Different coins and tokens call themselves "private" but just few of them follow a formal definition for this term. We introduce our statement of privacy and show that Cutcoin implementation of PoS protocol, including block ordering and validation, doesn't break it. One important feature of the protocol is that the block creation time can be significantly decreased compared to PoW-based coins. This lowers transaction latency and increases overall network throughput.

I Introduction

Establishment

In 2008 Satoshi Nakamoto published the manifesto "Bitcoin: A Peer-to-Peer Electronic Cash System" [1] that supposed to be a kick off to the modern cryptography-based financial system. The paper gave a short but exhaustive recipe of how several relatively simple technologies can be combined together to obtain a qualitatively new phenomena that was called 'Bitcoin', or cryptocurrency. They include digital signatures, distributed p2p database, or 'blockchain', the concept of 'consensus', or agreement between nodes in the

p2p network and a set of the rules of governance. Further on we will walk through the privacy aspects of different coins, the consensus algorithms pros and cons, and finally elaborate on the motivation for the new coin development.

Privacy

Comparing to fiat currencies, Bitcoin has obvious advantages: there's no trusted party it depends on; fast and transparent transaction processing and, among others, privacy. Privacy may be a requirement due to personal reasons, but it's also important as in the modern world the vast majority of payments made both by individuals and companies are processed in an electronic form. The corresponding information about transactions and the involved parties is confidential, however, it can be stored and accumulated by banks, authorities and other financial institutions. This data attracts attackers around the globe, sometimes they get unauthorised access and use it for fraud, fishing, etc, that evolves into financial and reputational losses [2, 3]. Traditionally, financial institutions protect clients' data by restricting access to their informational systems. There's also an alternative that lies behind cryptocurrency.

Some digital currencies let a person transferring the funds to stay anonymous, furthermore this guaranty is provided by the algorithmic methods. This option looks tempting for clients, and many companies claim their coin or token is private without specifying what exactly is meant. We assume that in order to achieve *hard privacy* [4] a digital currency must meet several conditions:

(i) anonymity, means that the parties involved in any transaction don't reveal any data that can help to identify them (IDs, number of participants, transferred amounts);

(ii) untraceability, or inability of coins (funds) origin tracing;

(iii) unlinkability, i.e. no any 2 transactions issued by any sender or received by any receiver can be associated (linked) by an observer;

(iv) implying of (i-iii) not only for fund transfers, but also for coinbase (mining) transactions and transactions to exchanges.

The properties (ii and iii) are difficult to achieve in practice and even harder to formally prove. However, during the years passed from the moment

when Bitcoin appeared on the scene, multiple cryptocurrencies have been developed and there are some attempts to solve the privacy problem in different ways and with the different degree of comprehensiveness. We refer to some of them.

Bitcoin approach. User identifiers are the public keys K of non-symmetry key pairs (k, K) such that $k*G = K$, the latter is a scalar product on elliptic curve [5]. Public keys are called 'addresses'. They can be used as an origin or destination in coin transfers, instead of real-life person identifiers. An observer would be able to see only funds transfer between different public keys K [1, 6]. Secret keys are required to subscribe a transaction when transferring funds and are not public therefore.

As an addresses can be traced in the blockchain and the payments from a single account can be easily linked, this approach implies only (i) from the list above. Furthermore, in case the person's IP address could be associated with the corresponding K their anonymity would be in danger. For this reason, Bitcoin is not fungible: newly minted coins are more valuable as they yet have no history in the blockchain. The techniques being used to overcome Bitcoin protocol vulnerabilities are centralized / decentralized mixing of a transaction participants. These have different realizations and provide different levels of privacy, but it's worth noticing that they are not built-in and that's a user's obligation to use them.

CryptoNote (CN) family. These coins, such as Monero or Bytecoin, have smarter algorithms and provide higher level of privacy [7, 8]. In CN [5] each user has two sets of private/public keys, (k^v, K^v) and (k^s, K^s). k^v is called 'view key' and can be used to determine if the coins addressed to a specific user and k^s is the 'spend key' which is required to send a transaction. Imagine the situation where Alice is willing to make a transfer to Bob and she knows his public keys (the address). She generates one-time public key K^0 using random value R and cryptographically secure hash function H:

$$K^0 = H(RK_B^v)G + K_B^s \tag{1}$$

Then Alice uses K^0 as the receiver address sending in addition RG in the payment transaction. When Bob receives the transaction he can calculate $k_B^v RG = RK_B^v$ (note that only those who know k_B^v can figure out who is the recipient of the payment) and check if it fit (1).

Transaction amounts are hidden using Ring Confidential Transaction technique [5], so the exact values are not visible for all except the sender

and receiver. Still other users can verify that the sum of input amount is equal to sum of output amounts. It's possible due to special *amount commitment*, it's included into a transaction by the sender and let others verify that the coins don't appear out of thin air.

From this example we can conclude that the users not only have anonymous identifiers, but the coins are also untraceable (ii) and the transactions to a single address are unlinkable (iii). A slightly bigger amount of actions required for the transaction to be completed – it is the price users pay for the privacy.

Both Monero and Bytecoin use Proof of Work (PoW) algorithm for consensus.

Zerocoin family approach differs a lot from the one used in CryptoNote. Zerocoin actually is rather a series of extensions (Zerocash protocol) to the existing Bitcoin blockchain than an independent platform [9]. To make a transfer Alice (sender) mint a specific amount of zerocoins by generating a random coin serial number S and then creates a *Pedersen commitment* c to S using random value R [10]. From the technical point of view it looks like a valid bitcoin transaction containing enough bitcoins to 'pay for' it. The commitment means that S can be revealed only knowing R.

After that Alice adds her token C to the blockchain so that any other participant can see it. Next, Alice creates a *non-interactive zero-knowledge proof* π for both of the statements:

(i) she knows a commitment $c \in (c_1, .., c_n)$ and

(ii) she knows R such that c opens to S.

Note, that she doesn't reveal which *exactly* commitment she knows. The proof reveals no information about Alice but let others verify its correctness. Finally, Alice publishes a transaction containing π and S with Bob's address as destination and with an empty sender address. After verification the amount of bitcoin equal to the zerocoin denomination is transferred from the zerocoin escrow pool $(c_1, .., c_n)$. Instead of sending the coins to Bob, Alice can redeem them by herself thus using zerocoin as a laundry service.

Zerocoin establishes high level of privacy in transactions as the chain of 'sender $\rightarrow$ receiver' addresses is untraceable. Still, it suffers from the issues peculiar to all proof of work backed currencies.

Anyway, we can conclude that mixing of the sender/receiver addresses, splitting of the public and private key pairs into several sets, zero knowledge proofs and special escrow accounts are useful techniques that can be

employed by cryptocurrencies to provide high level of privacy. Their correctness is proved from the mathematical point of view and their effectiveness is proved by massive practical usage.

Consensus algorithms

The absence of a single trusted party is definitely the strong side of cryptocurrencies. Instead of trusting to somebody participants achieve consensus on the history of the operations in the system. We don't discuss these mechanisms in details and generally follow this analysis [11].

First cryptocurrencies such as Bitcoin rely on Proof of Work (PoW) algorithm. If a node (miner) wants to add a new block into the blockchain they need to solve a difficult (resource consuming) task. One of them does it first, creates the block, publish it and receives the reward. Other network participants receive a block and before accept it can easily check that a certain amount of work has been done.

There may be a situation when different blocks come to different nodes at the same blockchain height, then alternative versions of the blockchain appear. There is a rule that allows one to solve this problem: any node accepts the longest alternative of the blockchain as the true version. In the most cases after several rounds of adding of new blocks nodes select one of the alternatives in the network.

The requirement of doing some 'work' protects the blockchain from attackers. Suppose some PoW based coin has the current blockchain height h_c. If an attacker wants to reorganise the blockchain at height $h_r < h_c$ they need to redo all the work that has been done to prove blocks from h_r to h_c plus one block on top of h_c. It was shown [1] that the probability such reorganization is high only in the case the attacker took control over more than 50% of the network's computational power (mining hashrate). It is called '51% attack' and there are several known cases when it was successfully performed [12]. As the result, transactions can be reversed and the same coins can be spent many times. It's an intrinsic behaviour of decentralized PoW coins; and it is partially the result of their nature: the chance to mine the next block is proportional to the computational power.

Other weakness of PoW based coins has its root in their high volatility. The miner's profit strongly depends on a coin's market price and block reward, the latter often decreases with time. While the prices for electricity and mining equipment can be thought as a constant, the price of coin itself may vary. When it goes down miners switch their computational resources to mine more profitable coins and 51% attack is likely to take place.

Pronounced bear market in 2018-19 when Bitcoin's (and many other coins') market price felt down made mining close to be unprofitable and many miners left this area.

The situation with block rewards is also unfair. The difference in computation power between common CPU/GPU and specialized mining device is huge, so ordinary miners may never find any blocks. They quit mining and computational power tend to concentrate in big pools managed by companies, thus making blockchain vulnerable.

In [11] several other reasons to switch from PoW are discussed. In the recent years new consensus algorithms were developed [13, 14]. Good candidate to substitute PoW is *Proof of Stake* algorithm. It uses *stake* to determine who will participate in creation of the next block. The stake means a commitment to other participants that a staker has some amount of coins enough to create a new block. PoS employs a deterministic solution to define the new block creator, and the chance that an account is chosen depends on its stake size. In PoS the blocks are said to be minted, or forged, rather than mined.

Motivation

There are lots of cryptocurrencies that claim to be 'private', but it is just a few of them to have privacy that is proved by the cryptographic apparatus used to conduct transactions. Monero coin looks most solid amongst them so we chose it as the base for Cutcoin. However, Monero miners have to mine new blocks using expensive equipment, following the mining algorithm updates and spending electricity. The idea of using PoS as the consensus algorithm for transactions looks promising. So all that being said, Cutcoin is the first private cryptocurrency with Proof of Stake consensus algorithm. The most challenging problem of a stake verification and preserving of privacy at the same time was successfully solved, further on we provide technical details.

II Cutcoin description

Definitions

The process of forging of new blocks is called ***staking***.

Block B_i is the basic element of Cutcoin blockchain at a height i.

In regard to subject area we widely use well established Monero terminology. The *unspent output* is such an output of a transaction that is not

yet an input of another transaction. The unspent output that is used to subscribe a forged block is called ***PoS output*** o^p. Unspent outputs contain *key image*, a 256-bit value used to validate transaction and prevent double-spending. Any output o has its intrinsic ***amount*** of coins a, their pair is (o, a).

PoS hash $h_i^p = H(o_i^p, h_{i-1}^p)$, where H is Keccak-256 hashing function [15].

Crypto hash $h_i^c = H(B_i)$ is the entire block hash.

Difficulty d is proportional to the total amount of coins currently being on stake.

Merkle root is the root hash of the transactions Merkle tree [5].

Each block has a single ***PoS transaction (ptx)***, it's a special transaction used to provide the participants information required for proving a block creation correctness.

Block ***forging*** is the process of a block creation in PoS systems.

In most cases we use i index to numerate blocks and j for outputs.

Blockchain structure

Cutcoin has the blockchain structure similar to one in CN coins [5].

The blockchain starts with the first block B_0 that has a genesis transaction. Following blocks are linked to B_0 one by one in a chain. In Figure 1 the fragment of Cutcoin blockchain is presented. Block B_i consists of the header and body. The header contains meta-information including PoS hash h_i^p and crypto hash h_i^c and thus linked to B_{i-1} as

$$h_i^c = f(h_{i-1}^c), h_i^p = f(h_{i-1}^p). \tag{2}$$

Note that

$$h_i^c = f(h_i^p), h_i^p \neq f(h_i^c). \tag{3}$$

PoS hash of B_i doesn't depend on its content, only on the previous block's h^p and unspent output of this block signer. It was intentionally designed this way because otherwise malicious users could 'mine transactions' to increase their chances to forge a block. From the technical point of view when a user signs B_i block his wallet generates special kind of transaction called 'PoS transaction'; it contains the signer's unspent output, persists in the block along other transactions and thus participates in the generation of cryptographic block hash.

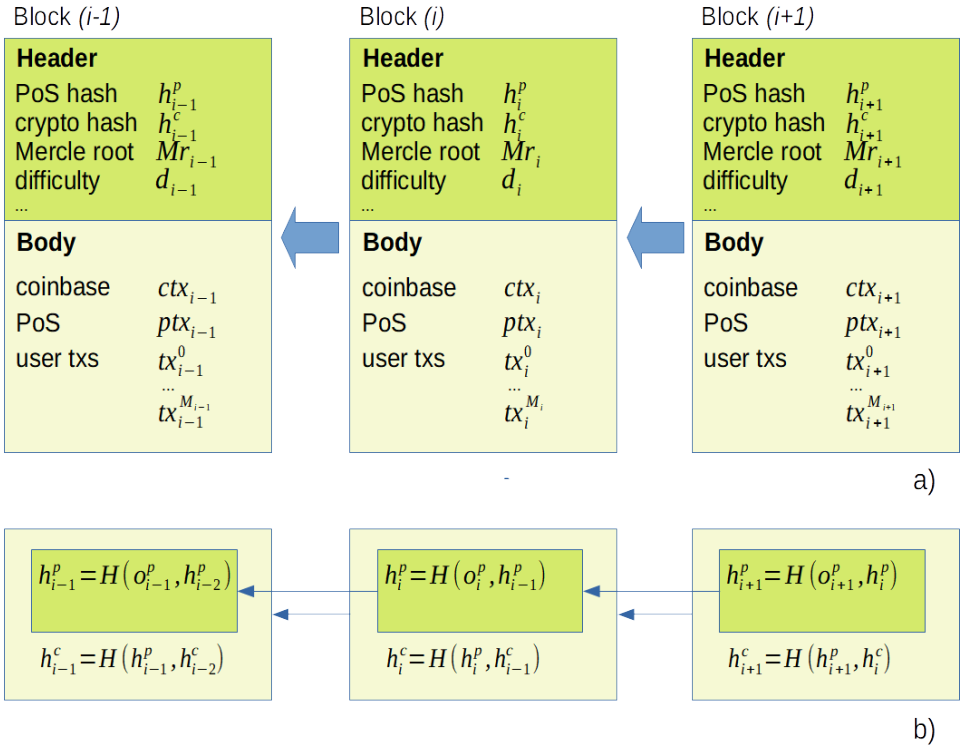

Figure 1: Fragment of Cutcoin blockchain: structure (a) and the corresponding chain of hashes (b).

Any changes in a block lead to change of its cryptographic hash h^c; due to (2) it causes change of hashes $h^p_{i,i>i_0}$ and require rebuilding of all the blocks following B_i and resigning of them. As the signers are distributed, independent and randomly chosen the probability of such reorganization is negligible. This protects Cutcoin blockchain against double spend attacks: no one transaction can be seamlessly modified.

As mentioned above, PoS transaction contains the unspent output signing the block that contains it, so a daemon can traverse the blockchain and verify that the PoS hash of each block was generated using the unspent output from its PoS transaction.

Cutcoin PoS model

Cutcoin PoS protocol is based on well known and reliable Nxt Proof of Stake protocol [16]. As discussed in one of Vitalik Buterin's papers dedi-

cated to PoS pros and cons [17], the main concern of consensus algorithms is blockchain resistance to any kind of attackers. He describes several major types: (i) denial-of-service, (ii) strategies that increase chances of a specific miner to mine blocks, (iii) medium-length forks etc. Their feasibility depends on the prediction of who could be the next (or i-th) block miner (determinism). Then the forging process can be manipulated. Reasonable solution, in this case, is introducing of randomness for choosing of an account that will forge a block.

Unspent outputs used to sign blocks are pseudorandom values that appear as a result of Cutcoin transactions. The choice of the next forging unspent output would be deterministic for a person who knows all unspent outputs in Cutcoin network. As the previous block PoS hash is also known, the output that will forge the next block can be easily determined. But in reality each user sees only his own outputs. Thanks to privacy protection mechanisms developed by Monero there's no means to obtain complete information about who owns which outputs and which of them are currently on stake. We can say, that from a single user's point view outputs to sign blocks are chosen pseudo-deterministically. This approach provide good resistance against different types of attacks [19] and resolves most of the concerns listed here [17].

Another requirement is fair distribution of rewards, i.e. not "rich get richer" [18]. Randomness in choosing of the output to sign a block is regularised so that the reward is strictly proportional to amount of funds being on stake. Each block is forged (and signed) by a single o^p belonging to the user who stakes and receives the corresponding reward.

Let's formalize the considerations discussed above: introduce the target function that has its minima at a different time for different stakers:

$$\mathcal{F}(t, a, o) = T(o, h^o) - aT_b t \frac{1}{\pi}\left[\frac{\pi}{2} - arctg(t_0 - t)\right], \tag{4}$$

here $T(o, h^p)$ stands for target value ($T_{ij}(o_j, h^p_{i-1})$ is the target value for specific o_j and h^p_{i-1}),

$$T = rand(o, h^p). \tag{5}$$

$T_b = K_s/d$ is the *base target* derived from the current network difficulty, K_s is the scaling factor; t is the time since the last block was created and t_0 is an average time between two sequential block creations.

We state that B_i is forged by a staker's output when

$$\mathcal{F}_{ij}(t, a, o_j^p) = 0 \cup t = \min_{j=1..N}(t_j) \tag{6}$$

In other words, the output that can provide $\mathcal{F} = 0$ equality first is used to sign the B_i block.

Now let's extract the time dependency from (4):

$$F(t) = t\left[\frac{\pi}{2} - arctg(t_0 - t)\right], \tag{7}$$

It has linear and nonlinear components. The linear one provides stable growth of the probability that the next block is mined and the nonlinear concentrate this probability in the area around t_0, see Figure 3. Such form provides guarantee that at some point of time block will be definitely created in spite of the PoS participants have no information about how mush coins are currently being staked.

With (7) we have

$$\mathcal{F}(t, a, o^p) = T(o^p) - aT_bF(t). \tag{8}$$

Note that the probability of B_i to be forged grows linearly with the amount a for the selected o_j.

Lemma 0.

$$(F_n(t) = 1/K_nF(t) \tag{9}$$

is the normalized cumulative distribution function (CDF) of the block forging with the normalization coefficient $K_n : \{0 \leq 1/K_nF(t) \leq 1\}$ (figure 3, blue curve).

$$P_f(t) = \frac{dF_n(t)}{dt} = \frac{1}{K_n}\left[\frac{1}{2} + \frac{t}{\pi((t_0 - t)^2 + 1)} - arctg(t_0 - t)\right] \tag{10}$$

is the probability function for the block B_i to be forged in a time t since the block B_{i-1} creation (figure 3, red curve).

Lemma 1. Outputs (o_j, a_j) with equal a_j have equal probability to forge B_i.

Assume N outputs (o_j, a_j) are on stake (we don't bother about which users they belong to for now) and $a_j = const()|_{j=1..N}$. We can think of $T(o_j)$ as of uniformly distributed random value $r \in Z : \forall r(0 \leq r \leq 2^{64})$.

Statement (4) can be interpreted so that for any o_j there is a time point t^p such that (6) is satisfied and B_i can be forged.

Any output has chances to create the new block B_i:

$$\Omega_i = \{r_1, r_2, ..., r_N\}. \tag{11}$$

As $a_j T_b F(t) = const()|_{t=t^p, j=1..N}$ (4) the i-th block will be created by the one which provides

$$o_{ij} : T(o_{ij}) = \min_{\Omega}(T(o_j)). \tag{12}$$

In (11) all events have equal probability of realization thus

$$P(o_{ij}^p) = \frac{1}{N}\Big|_{j=1..N}. \tag{13}$$

Lemma 2. If a user u has M outputs on stake $(o_m, a_m), m = 1..M$, $A_j = \sum_m a_m$ with $a_m \neq const()|_{m=1..M}$ the probability to forge B_i with at least one of these output is $P(o_{ij}^p) = A_j/A_\Sigma$ and equal to probability of forging with the single output (o_j, A_j). Here $A_\Sigma = \sum_{j=1}^N a_j$.

Indeed, the sample space for $T(o)$ is $\Omega_{64} = \{0, 1, 2, ..., 2^{64}\}$. $T(o_j)$ are i.i.d. random values, correspond the outputs held on stake by all users in the network and belong to narrower sample space $\Omega_N = \{r_1, r_2, ..., r_N\} \subseteq \Omega_{64}$. The realization $X_j = \{r_j\}$, so the probabilities of realization are

$$P(X_1) = P(X_2) = ... = P(X_N) = \frac{1}{N}. \tag{14}$$

Now consider the moment of time t_p when B_i is forged. Specific amount a_j is also random value independent on $T(o_j)$. In (8) time dependant part $F_p = const|_{t_p}$, so we can reorganize the forging condition:

$$\frac{r_j}{T_b F_p} = a, \text{ or } r' = a, r'_p = rk,\ k = \frac{1}{T_b F_p}, \tag{15}$$

k is the scaling factor depending on the difficulty in the network. The probability of realization of any output in the network is $\frac{1}{N}$ and the probability that it was any of the user's u block is $\frac{M}{N}$.

The probability of $r' \leq a$ is

$$\mathcal{P}(r' \leq a) = \int_0^a P_c(x)dx, \tag{16}$$

and P(x) is a constant in the corresponding range and otherwise equal to 0. Thus $\mathcal{P}(r' \leq a) = aP_c$. Now let's calculate, for instance, $\mathcal{P}(r' \leq a')$, $a' = 2a$. It's obviously

$$\mathcal{P}(r' \leq a') = 2aP_c. \tag{17}$$

And the probability of $\mathcal{P}(r'_1 \leq a)$ or $\mathcal{P}(r'_2 \leq a)$ is equal to

$$\mathcal{P}(r'_1 \leq a \cup r'_2 \leq a) = 2aP_c, \text{ and finally} \tag{18}$$

$$\mathcal{P}(r' \leq a') = \mathcal{P}(r'_1 \leq a \cup r'_2 \leq a). \tag{19}$$

The general case of m outputs:

$$\mathcal{P}(r' \leq ma) = \mathcal{P}(\bigcup_m (r' \leq a)). \tag{20}$$

Lemma 3. The output (o_j, a_j) being on stake has the probability to forge B_i equal to a_j/A_Σ, where $A_\Sigma = \sum_{j=1}^{N} a_j$, the total funds amount being on stake in the network. This statement can be proved similarly to Lemma 2.

The result of simulation presented in Figure 2 illustrates *lemma 3*. Here we model Cutcoin network with 1000 staking unspent outputs enumerated $\{o_1, o_2, ...o_{1000}\}$, each output of different denomination. The amount of coins that each unspent output contains grows linearly starting from 100 coins for o_1 with maximal 100000 coins for o_{1000}. During the process of simulation 1.0e6 blocks were forged and the scatter plot represents how many times each output was used to forge a block. As expected, the value fluctuates due to randomness peculiar to the process, however overall dependency is explicitly linear.

These properties of the target function allow persons flexibly stake their funds and guarantee fair distribution of rewards proportional to the stake amount.

Difficulty function

From (4) t_0 is the average time of B_i creation, in Figure 3 one can see that probability distribution function has its maxima in this area. This value must be a constant in time to provide reliability and accessibility for transactions. As the number of users and their coins being on stake changes from time to time $A_\Sigma \neq const(t)$ there should be a parameter that compensate these changes. This value is called base target, it is inversely proportional to difficulty. The difficulty is evaluated for each block using weighted smoothing:

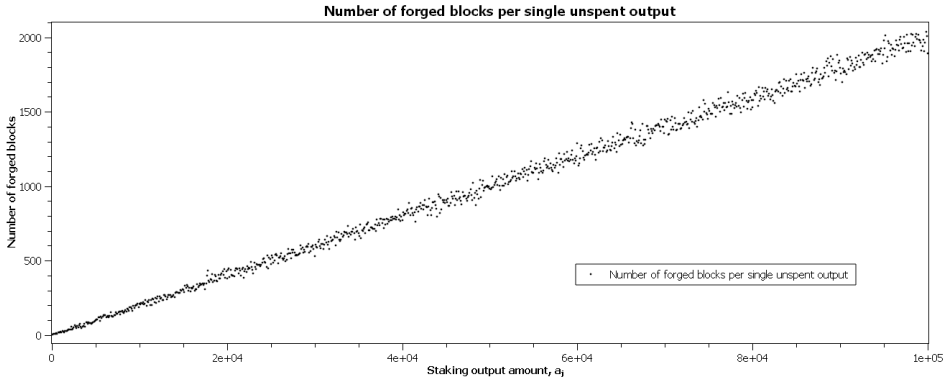

Figure 2: Number of forged blocks per single unspent output. Output's amount linearly grows.

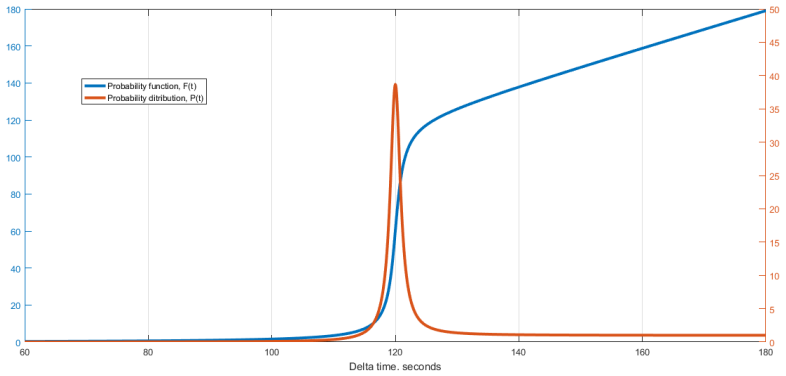

Figure 3: Cumulative probability function $F(t)$ and probability distribution $P(t)$ for a block to be forged.

$$d_i = \frac{\sum_{k=1}^{M} k d_{i-M+k}}{\sum_{k=1}^{M} k t_{0(i-M+k)}} \tag{21}$$

This form of smoothing provides reasonable combination of compensating behaviour in case t_0 starts to change rapidly and history accumulation. We use $M = 90$ blocks. c presents the results of simulation of Cutcoin network behaviour at rapid stake amount growth and drops. In the first case we increase the supply being on stake and monitor how the difficulty changes; in the second decrease it and do the same. The network remains

stable even though the amount changes 10 times from the original value.

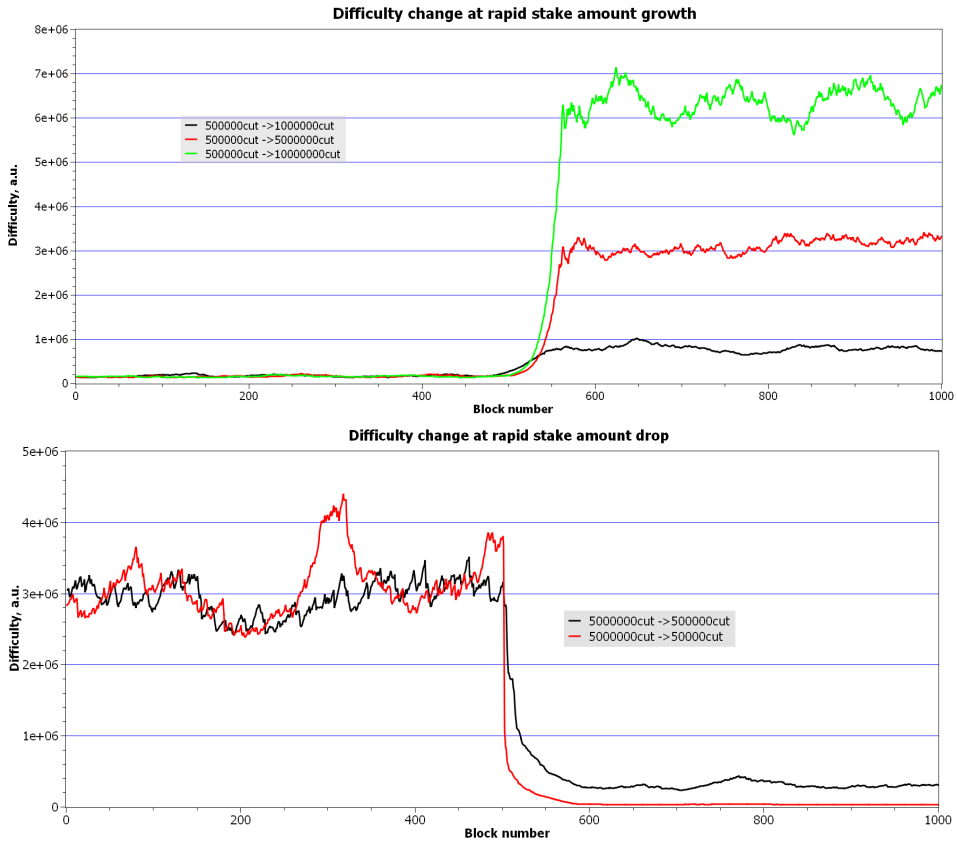

Figure 4: Simulation of the block generation process; difficulty change at rapid stake amount growth (a) and drop (b). Change happens between 499-th and 500-th blocks.

Realization details: forging algorithm from a single node point of view

Forging. As it was mentioned before, the major challenge of a private PoS coin is providing of the mechanism to prove forging correctness without revealing of private information to the participants. In Cutcoin it was solved in an elegant manner.

Suppose a person has N outputs $(o_j, a_j), j = 1..N$ with different amounts a_j on stake. At some time the person's wallet receives the message from the

daemon with the notification that the previous block B_{i-1} has been created. In the message the wallet receives h^p_{i-1} and use it to calculate $T_{ij}(o_j, h^p_{i-1})$ for each output. After that it can find the output that provide minimal time $t^p = \min(t_k)|_{k=1..N}$. This output is the *best try* for the wallet to forge B_i. Wallet builds it including PoS transaction serving several purposes:

(i) let the participants prove that o_j exist in the blockchain, is not spent or blocked;

(ii) let the participants prove that o_j has an amount $a_j : a_j > a^p$ sufficient for forging B_i at t^p.

There is also a special rule aimed to avoid attacks on the PoS hashes chain. An unspent output o_j must have at least 800 block confirmations, it is also checked.

PoS transaction does a simple thing: it transfers amount a_j to the same account with the zero fee. This transaction can be verified by any node's daemon using amount commitment and range proof (4) as a normal Monero transaction. It has exactly one input and two outputs, one is actual and another is dummy, artificially generated for privacy reasons and having zero amount, we don't take it into consideration. The following parts are equal:

$$x * G + a_{in} * H = y * G + a_{out} * H, \tag{22}$$

$H = H_p(G)$, a_{in} is the unspent output, a_{out} is an output, x and y are blinding factors (masks).

In PoS transaction we reveal $x * G$ and amount a so any participant can verify that the unspent output o_j had enough amount a_{in} to sign the block.

Distribution of block creation time probability. As we mentioned, (7) has nonlinear component. This part of the cumulative distribution function looks quite similar to Normal distribution's one:

$$\frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right], \tag{23}$$

and its derivative, the probability function $P(t)$ looks similar to Normal distribution. So the reasonable question is why not to use in (7) just normal distribution with the linear growth factor:

$$\mathcal{F}(t, a, o) = T(o, h^p) - aT_b t\left[e^{\frac{-(t-\mu)^2}{2\sigma^2}}\right] \tag{24}$$

The answer is related to the specifics of the realization. Any node in the network must be able to verify the legitimacy of the block creation and thus correctly and identically evaluate the forging time from (4). On the other hand different hardware platforms may have different floating point arithmetic realizations. Due to these reasons we used in-house 128-bit fixed point arithmetic library. Error function in (23) should be represented using exponents and this kind of functions is rounding errors prone.

There are several good approximation of arctangent in fixed point arithmetic so is was chosen to be part of cumulative distribution function.

Average block creation time. From (7) we can see that the average block creation time can be easily changed by just tuning of t_0 value, see Figure 5. PoW consensus algorithms need to keep this value big enough as it let miners to do much work and make it expensive for potential attacker to reorganize a blockchain. In opposite, PoS blockchain sustainability doesn't suffer in case of relatively small t_0 as it is protected by the random choice of unspent outputs to sign blocks. It looks like a big advantage as lets process much more transactions per time unit and decrease blatancies.

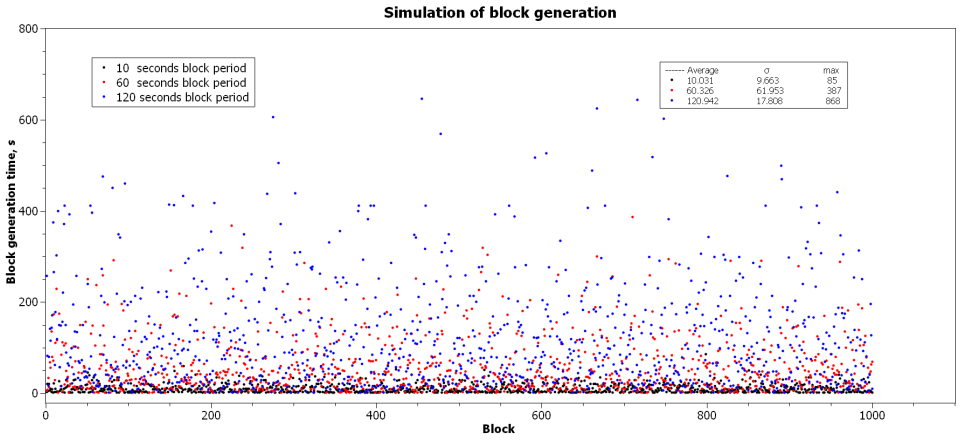

Figure 5: Simulation of the block generation process; block creation respecting different target times: t_0=10s, t_0=60s, t_0=120s.

III Conclusion

In this paper we have presented a new private coin based on Proof of Stake consensus algorithm. PoS coins are sustainable for various types of attacks

and provide larger functionality than traditional PoW ones. We proposed the randomisation mechanism for the choice of the next block miner. The achieved randomness is regularised so that the probability is proportional to the account's balance, that means block rewards have fair distribution between users. Another advantage is possibility of 'fast forging', that means the block generation time is about tens of seconds instead of minutes or tens of minutes, which is typical for PoW algorithms.

The proposed PoS algorithm and the elements of randomness used to choose the next block forger protect the blockchain from the several types of attacks, including 51% attack typical for PoW consensus algorithms. The system is successfully implemented and available for users as Cutcoin. We expect that in future Cutcoin network will enable various applications and services to run across a common layer of privacy.

References

[1] Satoshi Nakamoto. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Nikki Fiorentino, Kelly Dwyer, Alex Hamilton, Karen Barney, Megan Dwoskin, Eugenia Buggs, Laurel Laluk, (2018) The Impact of Cybersecurity Incidents on Financial Institutions, https://www.idtheftcenter.org

[3] Threat Analysis, www.mwrinfosecurity.com/assets/swift-whitepaper/mwr-swift-payment-systems-v2.pdf

[4] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, Wouter Joosen (2011), A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements, Requirements Engineering, 6(1), pp. 3-32.

[5] Kurt M. Alonso, (2018) Zero to Monero.

[6] Wai Wuv, Brett Falk, (2018) Limitations of Privacy Guarantees in Cryptocurrency.

[7] Nicolas van Saberhagen, (2013) CryptoNote v 2.0.

[8] Daniel Genkin, Dimitrios Papadopoulos, Charalampos Papamanthou, (2018) Privacy in Decentralized Cryptocurrencies, Communications of the ACM, Vol. 61 No. 6, Pages 78-88.

[9] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin, (2013) Zerocoin: Anonymous Distributed E-Cash from Bitcoin, Conference Paper.

[10] Alex Vazquez, (2019) ZeroCT: Improving Zerocoin with Confidential Transactions and more.

[11] Earls, J.The missing explanation of proof of stake version 3, (2017) http://earlz.net/view/2017/07/27/1904/the-missing-explanation-of-proof-of-stake-version.

[12] Congcong Ye, Guoqiang Li, Hongming Cai, Yonggen Gu, Akira Fukuda, Analysis of Security in Blockchain: Case Study in 51%-Attack Detecting, (2018) 5th International Conference on Dependable Systems and Their Applications (DSA).

[13] Giang-Truong Nguyen, Kyungbaek Kim, (2018) A Survey about Consensus Algorithms Used in Blockchai, JIPS, Volume: 14, No: 1, Page: 101 - 128.

[14] Peter Urban, Andre Schiper, (2004) Comparing Distributed Consensus Algorithms.

[15] Guido Bertoni, Joan Daemen, Micha el Peeters and Gilles Van Assche, (2008) Keccak specifications.

[16] Nxt White paper, https://nxtwiki.org/wiki/Whitepaper:Nxt.

[17] https://vitalik.ca/files/randomness.html.

[18] Giulia Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, Gerui Wang, (2018) Compounding of Wealth in Proof-of-Stake Cryptocurrencies

[19] Serguei Popov, (2016) A Probabilistic Analysis of the Nxt Forging Algorithm.