Proof of usage: user-centric consensus for data provision and exchange

Samuel Masseport, Jorick Lartigau, Benoit Darties, Rodolphe Giroudeau

▶ **To cite this version:**

Samuel Masseport, Jorick Lartigau, Benoit Darties, Rodolphe Giroudeau. Proof of usage: user-centric consensus for data provision and exchange. Annals of Telecommunications - annales des télécommunications, 2020, 75 (3-4), pp.153-162. 10.1007/s12243-020-00753-8 . lirmm-02541049

HAL Id: lirmm-02541049

https://hal-lirmm.ccsd.cnrs.fr/lirmm-02541049

Submitted on 12 Apr 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Noname manuscript No.
(will be inserted by the editor)

Proof of Usage: User-centric consensus for data provision and exchange

Samuel Masseport · Jorick Lartigau · Benoît Darties · Rodolphe Giroudeau

the date of receipt and acceptance should be inserted later

Abstract This paper presents a new consensus algorithm, Proof of Usage (PoU), for the blockchain technology. This consensus is introduced for permissioned (or private) blockchains and is designed for a user-centric personal data market. This market is subject to specific regulations with which conventional blockchains fail to comply. Proof of Usage aims to promote a new paradigm dedicated to usage incentivization, valuation, and control of user data in various sectors, such as banking and insurance. Other consensuses such as Proof of Stake or historical Proof of Work do not encourage coin spending and usage (in fact, Proof of Stake promotes the opposite). However, the value of the currency mainly depends on its use. This paper first introduces a contextualization of blockchain technology and decentralized consensus models. The motivation is then discussed for a new model of personal data exchange in a decentralized but supervised environment. The PoU protocol and its process flow are defined in detail. Furthermore, the paper explores two different approaches regarding the reward mechanism and the incentive model. Finally, the paper focuses on security requirements and how PoU meets such requirements in a permissioned-based blockchain system.

Samuel Masseport
Pikcio SAS, Montpellier, France
LIRMM, University of Montpellier, CNRS, Montpellier, France
E-mail: samuel.masseport@{lirmm.fr,pikcio.com}

Jorick Lartigau
Pikcio SAS, Montpellier, France
E-mail: samuel.masseport@pikcio.com

Benoît Darties
LIRMM, University of Montpellier, CNRS, Montpellier, France
E-mail: darties@lirmm.fr

Rodolphe Giroudeau
LIRMM, University of Montpellier, CNRS, Montpellier, France
E-mail: rgirou@lirmm.fr

1 Background

In November 2008, the first fully decentralized cryptocurrency, Bitcoin, was introduced by an unidentified person or group acting under the pseudonym of Satoshi Nakamoto [1]. To this day, Bitcoin is the most widely adopted and reliable economic system built on a peer-to-peer network, enabling online payments directly from one party to another without a sole fiduciary entity. This paper was the starting point for Blockchain technology [2]. The Bitcoin consensus is based on a Proof of Work (PoW) protocol [3, 4] that protects the network from denial-of-service (DoS) attacks and double-spending. New transactions are included by adding blocks to the blockchain. Each block is added by a single node determined by the system. In a PoW protocol, nodes (named "miner") compete to add a block to the current chain by trying to solve a hard asymmetric mathematical problem.[1] The Bitcoin blockchain rewards the miner when it adds a block to the current chain. Such an incentive model encourages miners to use their computing power to improve the defence of the network. However, PoW has two mains issues: the need for a significant amount of computing power and its limited scalability in terms of transaction processing (currently around seven per second on the Bitcoin network). Intuitively, a node with high computing power is most likely to be selected by the system. The rise of Bitcoin has led to the emergence of new crypto-currencies and the implementation of technologies to overcome the limitations of PoW.

As Bitcoin was often criticized for its high energy consumption due to the fierce competition to add a block, a new consensus protocol was proposed: the Proof of Stake (PoS) algorithm. It was introduced by Sunny King and Scott Nadal in 2012 [5]. Rather than using the computing power of miners to add blocks to the current chain and secure the network, King and Nadal defined an alternative method called "staking", in which a deterministic algorithm chooses a network participant (i.e., a node) to seal the current block. The selection is based on the number of coins in each node's possession or "stack". For example, if there are 100 coins on the network, a node holding 10 coins will have a 1-in-10 chance of being selected as the block miner and earning the reward. Nodes holding 10 coins are 10 times more likely to close and add blocks than nodes with a current stack of one. Thus, the more a node stacks, the higher the closing probability. This results in a model where nodes are encouraged to stack their coins rather than spend them.

New consensus models have appeared over time, proposing new incentive models and different objectives. For instance, the Proof of Activity (PoA) algorithm by Bentov et al. [6] is a merging of the concepts of both PoW and PoS with a redistribution of the gathered fees to a set of randomly selected stakeholders. In the PoA consensus, miners use their computing power to try to add blocks to the current chain state. However, the miner solving the PoW will not earn the full pool of fees. A fixed number n of stakeholders are selected

[1] An asymmetric mathematical problem is a mathematical problem that is hard to solve but the solution of which is easy to verify.

using the follow-the-Satoshi (FTS) algorithm,[2] using the hash of the previous block concatenated with n fixed suffix values as input. All selected stakeholders sign the block in a given timeframe, one after another, to prove their activity. The selected stakeholders are registered within the block. The fees assimilated as rewards are redistributed between the miner and the selected stakeholders. The PoA model incentivizes nodes to be fully available (i.e., connected) to sign new blocks and earn associated rewards.

An improvement of the PoA model could be to encourage spending over stacking. This could be a fundamental criterion for the sustainability of the currency. For instance, Proof of Importance aims to reward usage over stacking. It is a consensus algorithm introduced by NEM [7] and based on PoS and a scoring system. To be eligible to close a block, a node needs to have in its stack a minimum of 10,000 coins. Additionally, each node has a score that increases according to the number of coins in its stack and the number of transactions initiated within the last 30 days. The larger and more frequent transactions processed, the greater the impact on the Proof of Importance score. Thus, the closing node will be selected according to its score, which will be reset upon block submission.

Proof of Importance is a consensus algorithm that encourages nodes to exchange their coins through public infrastructure (i.e., a public blockchain), where any node has read and write access to the distributed ledger. Public blockchains are fully decentralized, meaning no single person has control over the network. Two examples of public blockchains are Bitcoin [1] and Ethereum [8]. Usually in public blockchains users are pseudonymized, which does not necessarily mean anonymity. Indeed, public blockchains record transactions in a fully shared, decentralized ledger that by definition cannot offer privacy of network operations [9, 10].

Permissioned blockchains (or private blockchains) work similarly but with access controls that restrict those that can join the network. Permissioned blockchains have a single or multiple entities who control the network, such as third parties who regulate user transactions. A well-known example of a permissioned blockchain is Hyperledger [11]. It should also be noted that consortium blockchains exist. They represent an intermediate model, in which the participating infrastructures are controlled by different entities, unlike private blockchains, in which a single operator manages the platform.

In this paper, we propose the Proof of Usage (PoU) as a consensus that encourages nodes (i.e., users or participants) to spend coins for service delivery. By using their coins, they sustain the usage of a stable currency over time in an environment that is both permissioned and distributed. This consensus is introduced for PikcioChain [12, 13]. The paper is organized as follows. The next section discusses the motivations and reasons to use a permissioned blockchain environment. Section 3 describes the protocol of the consensus, and Section 4 discusses different approaches to the reward system. The security of PoU is

[2] FTS is an algorithm that picks a coin and selects its owner as a leader. The more coins a node has, the more likely it will be selected.

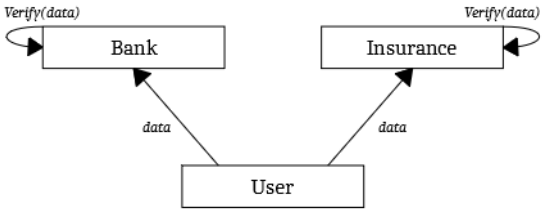

Fig. 1 Model of data exchange between bank, insurance, and user.

presented in Section 5, after which the conclusion provides a summary of what has been accomplished and a discussion of future research directions.

2 Motivations

Personal and identity data are the backbone of any service delivery, especially in the digital sphere. Each time a customer wants to take out a loan (Fig. 1), he or she must provide a substantial amount of personal data to the bank. Furthermore, banks usually require a complete identity evaluation using the customer's data (identity card, name, age, address, etc.). The data are processed, crossed, and finally validated to establish the customer's digital identity. When this same customer applies for insurance cover, however, he or she will be required to provide all the information again because in most cases there is no direct communication link between the entities (bank and insurance firm). As a result, both the bank and the insurance provider check the data before validating the customer's identity. This process is both time-consuming and costly, with no particular return on investment for companies, which leads to poor customer experience.

On the other hand, data, especially personal data, have become a vast market powered by mass adoption of digital services. As a result, service providers buy large amounts of data to try to attract new consumers. Generally, the data are bought from third parties, such as social networks or email providers, and the rightful owner (i.e., the user) is completely excluded from the transaction and cannot apply any sort of control. Since the user's consent is not required (Fig. 2) or drowned in over-complicated terms of use, current data business models do not include users, despite them being the provider of the value. Only data sellers or advertising agencies earn money from users' data. To develop a more user-centric model, giving users control over their data, two directives can be proposed:

1. Data sellers, such as social networks, redistribute a part of their incomes to their users, which means a reduction in their profits.
2. The data valuation costs payed by data buyers, such as advertising agencies, are increased to reward their users.

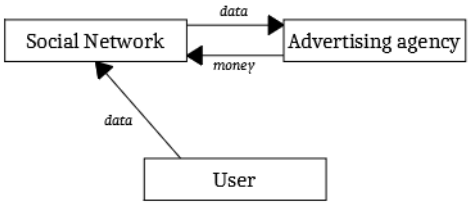

Fig. 2 Model of data exchange between a social network, an advertising agency, and a user.

Such a model could be described as Utopian. If such a model can be proven to be both a carrier of more qualitative data and user-controlled in accordance with current regulations, such as the General Data Protection Regulation [14], it could enable a more virtuous data-exchange process.

The model proposed in this paper (Fig. 3) is a user-centric model of data exchange between several entities (e.g., banks and insurance providers) in which the user can control the sharing of his or her data. This model is an alternative to the model given by Fig. 1 and is based on the model proposed in Fig. 2 and empowered by the user's control of his or her own data. In this model, a user still has to give his or her personal data to the bank to take out a loan, and the bank still has to use the data to verify the customer's digital identity. However, when the same user applies for insurance cover, the insurance agency can request the certified data from the bank in exchange for payment (upon the user's approval) that is redistributed directly to the user. In such a model, companies (e.g., banks and insurance providers) can access certified data (after purchase), which involves less processing cost and more importantly saves time, leading to a better user experience. The company that processed and certified the user's data now receives a return on investment for their validation work. Moreover, directly including the user in the authorization process and giving the user control over the transactions with his or her personal data is a strong trust marker in the customer-business relationship. Such processes can be fully automated and decentralized in a consensus protocol. Hence, PoU is introduced to both incentivize usage and create new benefits for data owners (i.e., private individuals), enabling a more virtuous business model, a more trust-based relationship, and increases the value of the data (i.e., through the certification).

Note that this model is only viable in a permissioned blockchain, given the holding of sensitive information within the distributed ledger technology. The General Data Protection Regulation aims for more effective and more virtuous data governance with which public blockchains cannot comply (see the General Data Protection Regulation [14] for more details).

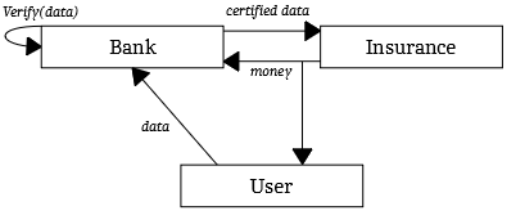

Fig. 3 Our model of data exchange between a bank, an insurance provider, and a user. "Certified data" are the personal data of the user that have been verified by the bank.

3 Description of the protocol

Proof of Usage can be regarded as an extension of the work of Bentov et al. [6] on PoA, where users are incentivized to exchange coins rather than simply be connected. This section describes the PoU process flow. Proof of Usage is based on the PoS selection principle. However, its scope does not extend to the global supply and dispatch of coins to the stakeholders; rather, it concerns the overall number of exchanges processed within the last block of the current chain.

In a permissioned blockchain, joining the network requires access to be granted by administrators, which significantly reduces the risk of external attacks on the system but does not completely remove it. In fact, before becoming a participant of the blockchain, a party must complete a full identification process, regardless of whether the party is a private individual or a company. Administrators can then accept or refuse the party based on the identification process, which includes risk analysis. Indeed, in the case of PikcioChain, the network relies on a know-your-customer (KYC) process [15] that enforces the international anti-money-laundering laws [16]. Participant and validator access is therefore restricted and controlled. A given entity cannot create multiple identities to take over the network governance.

On the network, there is only a fixed number of validators, called masternodes, which verify transactions and earn rewards for adding blocks to the chain. Only masternodes share the ledger and are therefore the sole entities with writing rights. Any node can become a masternode by making a request to the administrators with an amount of coins to pawn. However, as there is a fixed number k of masternodes, not everyone can be a masternode at the same time, only the k nodes that proposed the most coins will become masternodes (as in an auction). If a masternode tries to corrupt the network, administrators can replace it with another, and its pawned coins will be removed. If a masternode no longer wants to be a masternode, it must send a request to the administrator to be replaced and retrieve its coins. Otherwise a masternode remains a masternode as long as it exhibits "good" behaviour towards the network's best interests (i.e., as long as it performs reliably). The number k depends on what the administrators want to prioritize on the network. The

larger k is, the more decentralized and slower the network; the smaller k is, the more centralized and faster the network. To initialize the network, the administrators generate the k masternode addresses, each one corresponding to a 256-bit number.

Consider a set M of masternodes (numbered by $m_0, \ldots, m_{|M|-1}$) as validators and a set U of nodes. Additionally, connections between masternodes form a complete graph (i.e. each masternode is connected to all the others).

Blockchain can be seen as a state transition system in which a state consists of the ownership status of all existing tokens or coins and a state transition function that inputs a state and a set of transactions and outputs a new state. Let S be a state of the blockchain and $changeState(S, T) = S'$ a state transition function that takes as input S with a set of transactions T (numbered by $t_0, \ldots, t_{|T|-1}$) and outputs S' as the new state. For example $changeState(\{A$: 25 tokens; B: 10 tokens$\}$,$\{A$ sends 20 tokens to $B\}) = \{A$: 5 tokens; B: 30 tokens$\}$

In PoU, the *changeState* function is defined as follows:

1. **Masternode selection:** select $m_s \in M$, the selected masternode that will add the next block B to the current chain.
2. **Transactions selection:** m_s selects some transactions in its current pending transaction stack to build a set of validated transactions T_B.
3. **Lucky nodes selection:** m_s selects "lucky nodes" - the nodes (i.e., users) that will earn the fees from the current block.
4. **Block sharing:** m_s creates the block and adds it to the blockchain to broadcast it over the network.

The *changeState* function is a cyclic function executed at regular intervals. In the following parts, the SHA-256 hash function is used. This function returns a 256-bit number.

The following sections describe the different protocol phases.

3.1 Masternode selection

Masternode selection is a process that consists of selecting the masternode m_s ($\in M$) that will create, sign, and add a new block to the blockchain. To select m_s, the hash of the previous block is compared to the set of masternode IDs (Fig. 4). The masternode with the ID closest to the hash from the previous block is selected to be the next validator (i.e., the masternode). Therefore, m_s seals and pushes the next block. Note that all masternodes select the same m_s without broadcasting it, as each of them knows the hash of the previous block and the set of masternode IDs.

As shown in Fig. 4 masternodes do not have the same probability of being selected even if the hash function is evenly distributed. For example, here, m_1 is less likely than m_0 or m_2 to be selected. This should not be regarded as an issue, since masternode governance is auctioned. The administrator gives the address with the greatest chance of being selected to the masternode that

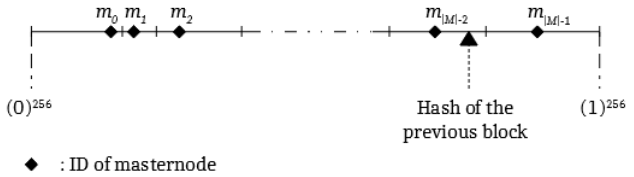

Fig. 4 Masternode selection. In this example, $m_{|M|-2}$ is selected to seal and add the current block. $(0)^{256}$ and $(1)^{256}$ respectively correspond to the lower and higher bounds of the output of the SHA-256 hash function.

has pawned the most coins. As a result, masternodes that have pawned more coins have a higher probability of being selected than masternodes that have proposed fewer coins.

As mentioned previously, the *changeState* function is a cyclic function executed at regular intervals. If the selected masternode m_s does not share a valid block after a fixed number of executions, the function will consider it unavailable. The masternode selection process will omit its ID in the next execution. This condition prevents the network from inflating in a pending state where m_s is not available for any reason (disconnection, network latency, system failure, etc.).

3.2 Transactions selection

In this work, we consider PoU as a supplier of two distinct types of transactions: classic transactions Tx and data transactions Tx-data, as depicted in Fig. 5.

1. A *classic transaction* is a transaction in which node A gives a certain number of coins or tokens to node B.
2. A *data transaction* is a transaction of monetized data using the system coins; for example, business A purchases in coins the certified data of user C from another business B upon the user's consent (i.e., A cannot initialize the transaction until C signs it).

When a node initializes a transaction, it sends its request to all masternodes. When a data transaction is initialized and accepted by the data owner, it will remain pending until the node that paid the coins receives the certified data. This pending state prevents malicious nodes from trying to receive payment without sending the data.

Each masternode stores locally a transaction stack that contains the transactions waiting to be written on the chain. When masternodes receive a request, they make sure the transaction is valid by verifying the node signature and the current balance in hold. If the transaction is validated, the masternode adds it to its current transaction stack.

When the system selects a masternode m_s to create a new block B, m_s adds transactions from its transaction stack to the set T_B for the current

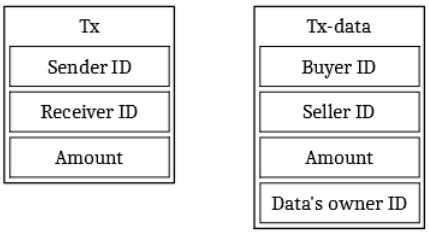

Fig. 5 The transactions model used in PoU. A classic transaction Tx (left) includes only a sender, a receiver, and an amount; a data-transaction Tx-data includes a buyer, a seller, an amount, and the identity of the owner of the data.

Transactions

Sender /buyer	Transactions type	Receiver /seller	Amount in coin	Data's owner
A	Tx	C	15	-
B	Tx-data	D	13	E
C	Tx	B	4	-
A	Tx	B	10	-
D	Tx-data	F	8	C

Fig. 6 Example of set of transactions of a block.

block (see example in Fig. 6). Ideally, m_s must process transactions in the order they are received by its stack (as in a queue). The inability of others to verify this (because the communication is asynchronous) cannot be regarded as an issue, as the order does not affect the system and transactions omitted by the selected masternode m_s can be included later by another masternode. For various technical reasons and to limit the overall size of a block, the number of transactions in a block is restricted.

3.3 Lucky nodes selection

As in the PoA protocol, PoU selects some nodes (i.e., users), called lucky nodes, to be given a share of the adding block reward. In PoA, a selected node needs to be online when the block is added to receive the reward. In PoU, the user (or the data owner in the case of a data transaction) must be at the origin of a transaction in the added block. Let us consider a set of n lucky nodes that shares the reward of the block with the selected masternode m_s. Selection of the n lucky nodes is based on the FTS algorithm. To use the FTS algorithm, consider that each coin of the network is signed and traceable. The FTS algorithm in PoS is used to select a node by selecting a coin in the network supply. The owner of the selected coin is the lucky node. In PoU, masternodes generate n hashes from the hash of the previous block concatenated succes-

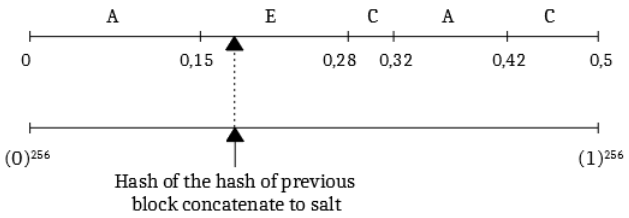

Fig. 7 Example of one lucky node selection in the block from Fig. 6 with transaction fees of 1%. The first line corresponds to the transaction fees of the block from Fig. 6 added one by one with the initiator of the corresponding transaction (or the data owner for a data transaction). $(0)^{256}$ and $(1)^{256}$ respectively correspond to the lower and higher bounds of the output of the SHA-256 hash function. Here, node E owns the coin determined by the hash of the hash of previous block concatenated with a salt, and it receives the transactions fees.

sively with n salts (known by all masternodes).[3] These hashes are scaled to the total transaction fees of the current block. The n hashes are compared to the transaction fees to select n coins and then n transactions. The n nodes that have initiated these transactions are the lucky nodes and earn a share of the reward. Note that a transaction can be selected more than once, and the same node can earn several shares. As the selection of lucky nodes is deterministic and all masternodes know the hash of the previous block and the salts, each masternode selects the same lucky nodes without communicating and clogging the network.

Regarding the number n of nodes to be selected, in this particular case, PoU selects a fixed number of nodes (one or more) or a number scaled according to the block's number of transactions or the total amount of exchanged coin in it.

Let us consider the selection of one lucky node from the previous block given by Fig. 6 and a transaction fees of 1% of the total amount in the transaction stack T_B. The lucky node selection is performed as shown on Fig. 7. The graduated line on the top represents the transaction fees of the block (i.e., 1% of the amounts of each transaction of the block given by Fig. 6). The hash of the hash of the previous block concatenated with a salt is scaled to the sum of the transaction fees to select a transaction. The initiator of the selected transaction (or the data owner in the case of a data transaction) is the lucky node. In this particular case, node E is selected and receives the transaction fees as a reward for using the service.

3.4 Block sharing

At this stage, m_s has validated the transactions in its stack to create the T_B set to include in block B. Now, m_s builds the header of the block, containing

[3] In cryptography, a salt is data that are used as an additional input to a one-way function that hashes data. Salts are used to safeguard passwords in storage, but here it is used to generate multiple hashes from a single input.

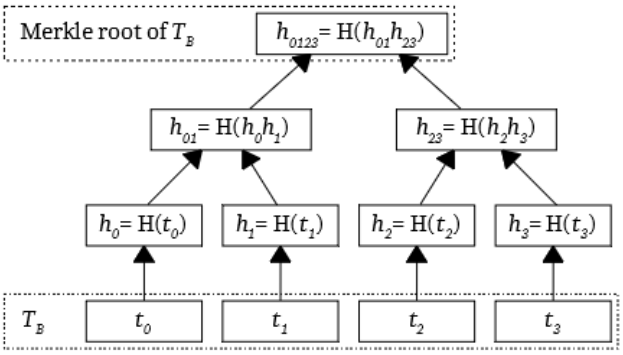

Fig. 8 Construction of the Merkle root of T_B. The function H() corresponds to any hash function.

the hash of the previous block (i.e., *Hash(B − 1)*), the Merkle root [17] of T_B (Fig. 8), the ID list of selected lucky nodes, and the timestamp of B. Finally, m_s signs the header with its private key to lock and secure the header. Note that the hash of a block B corresponds to the hash of its header concatenated with T_B and with the hash of the previous block $B - 1$.

At this point, m_s has successfully sealed the block B and added it to its chain (Fig. 9). It then broadcasts the new block to all other masternodes. When a masternode receives a block from another, it must validate its header and its content before adding it to its chain. A receiving masternode m_r considers a block B as valid if and only if the following conditions hold:

1. the previous hash in block B corresponds to the hash from the last block of the current chain of m_r,
2. the ID of the selected masternode m_s is indeed the closest ID from the previous block hash (i.e., m_s is the rightful validator),
3. all transactions within B (i.e., all transactions of T_B) are valid (there is a signature and sufficient funds), and
4. the selection of lucky nodes is not biased (i.e., m_s rewards the rightful nodes).

After receiving a valid block, m_r adds it to its chain and deletes processed transactions from its transaction stack. Condition (1) cannot be fulfilled in the following two cases: (i) m_s tries to share a block with false information and (ii) m_r is not up-to-date and its last block is an older block in the chain of m_s. Case (ii) can occur if m_r did not receive previous block(s) or if it reconnects to the network after some inactivity or failure. If condition (1) is not met, m_r proceeds to a recovery protocol. Let us consider the hash h of the last block B of m_r. The recovery protocol proceeds as follows:

1. m_r requests the hash of the current block from all masternodes.
2. All masternodes send the hash of their current block to m_r.
3. m_r selects the hash with the most occurrences h' and compares it with h. If $h = h'$ the masternode in recovery is up-to-date and the protocol

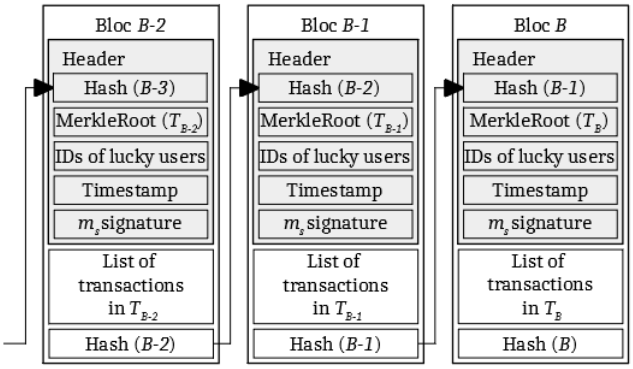

Fig. 9 Structure of the blockchain.

is terminated (corresponding to case (i)). Otherwise, the recovery process continues as follows.

4. m_r requests all the missing blocks starting from its B from any masternode with rightful last sealed block (i.e., any masternode that has sent h'). The blocks are numbered, and m_r can request the blocks between block B and the last block (corresponding to the hash h').
5. m_r validates block by block from B to the last block, as described previously.

This protocol allows m_r to rebuild the missing parts of the ledger from its last-known block to the current block of the chain. Note that the blockchain is up-to-date when at least most of the masternodes are up-to-date. This safety properties is a direct consequence of the recovery protocol. When masternodes that are not up-to-date execute the recovery protocol, they will request the hash of the current block and they will be updated according to the hash that appears the most often.

4 Transaction fees

This section explores the reward and incentive model; that is, how to incentivize masternodes to process transactions and blocks and reward lucky nodes selected within the current block. Rewards are issued directly from the selected masternode to the lucky nodes.

4.1 Model with transaction fees

Common reward systems rely on transaction fees. When a node spends coins for a given transaction, a percentage of the transaction amount is extracted as a reward. Thus, every time the selected masternode m_s adds a block, it earns a percentage of the total reward of the current block and equitably redistributes

the rest to the lucky nodes. Such a system usually works with a deflationary currency, but it can be adapted to an inflationary currency by generating or minting new coins for rewards.

The main problem with systems based on transaction fees concerns "classic nodes" and private individuals, who are not necessarily familiar with such models and crypto-currency ecosystems in general. In the common fiduciary bank model, when a user spends 10 coins, the recipient receives the whole sum and therefore does not expect to pay additional fees for the service.

4.2 Model without transaction fees

Another reward system involves creating the reward. In this system, fees are calculated as described in the previous approach as a percentage of the transaction amount. Note that such a system is inevitably inflationary, but when a node A exchanges 10 coins to another B, A pays 10 coins and B receives 10 coins.

However, this approach cannot prevent nodes from spamming large amounts of transactions across multiple accounts. Hence, such malicious nodes will not spend any fees and could try to participate in the lottery during each block's sealing process.

A solution is to limit transactions for each node over time, as in any conventional banking system. This limit can be defined according to the number of transactions and the total amount allowed of transactions within a given timeframe. If a node spends more than the authorized limit, transaction fees will be withdrawn. This limit prevents malicious nodes because they will start to spend fees that they may never retrieve.

From our point of view, minting new coins based on a transaction-fee system with limit control is an approach that can improve the usage and understanding of private individuals. This system is a balanced compromise between accommodating "common users" who do not want to pay fees despite their low usage and tackling nodes that are trying to corrupt the system.

5 How PoU deals with security threats

This section describes how PoU prevents attacks from malicious nodes to build a resilient and reliable consensus. Each of the following four subsections describes one type of attack.

Note that attacks from masternodes are not beyond the scope of this paper: such nodes are normally trusted, and the risk of an attack from them is reduced by several mechanisms (KYC; i.e., the administrator knows the real identity of masternodes, pawned coins, etc.). Moreover, PoU is designed for permissioned blockchains only and has strong user authentication to engender trust (users cannot hide behind pseudonyms). As a result, a given entity cannot create multiple identities on the blockchain.

5.1 Denial-of-service attack

A denial-of-service (DoS) attack, or flooding attack, involves rendering a machine or network resource unavailable to its intended nodes by flooding it with superfluous requests. Distributed DoS (DDoS) attacks are an extension of DoS attacks in which the targeted resource is flooded from many different sources, making it impossible to stop the attack simply by blocking a single source. Both DoS and DDoS attacks have become one of the main Internet security issues [18] and represent a serious cost to the online industry.

Such attacks to a blockchain ecosystem mainly involve flooding the network with dummy transactions to increase the verification time of transactions and blocks. The KYC system, however, prevents a user from creating multiple identities (and hence nodes) and prevents a DDoS attack (unless a group of nodes choose to unite). Moreover, the KYC system enables the administrator to know the identities of all real users and block the malicious nodes from performing DoS attacks (masternodes can easily spot and blacklist the node). In this case, the node is excluded from the network and loses its coins.

5.2 The problem of double-spending

Double-spending is a problem in which the same digital coins can be spent more than once. Double-spending is closely related to the 51% attack, in which a malicious node can rewrite the ledger if it has enough power on the network. If a node (or a set of nodes) gathers more than 50% of the writing capacity of the network, it can write on the blockchain faster than all other nodes of the network and thus rewrite the chain from a past block to a new current block. In this case, all transactions between this past block and the new current block are invalidated.

In PoU, the writing power is possessed by masternodes. They are the only ones that can launch a double-spending attack. In this section, we assume that masternodes can be trusted. However, what happens if one (or a small number) of them is malicious? Masternodes earn coins by adding valid blocks to the chain, and they can be banished and replaced if they fail to fulfil their goal. Indeed, a masternode loses coins by not adding blocks, leading to it being banned from the masternode pool. As long as more than 50% of the masternodes are trusted (and up-to-date), a double-spending attack cannot occur.

5.3 Power of the users pool

Generally, distributed systems suffer from groups of nodes (called a "pool of nodes") that join forces to increase their power over the blockchain, depending on the type of consensus in place. For example, in PoW, the power of a node is its computing capacity. As in PoS, the power of a node depends on the number

of coins in each node's balance. In PoU, nodes that have the power are nodes that exchange coins. Indeed, to increase their chances to being selected as lucky nodes, nodes need to increase the total amount of their transactions. Thus, if nodes of a pool want to be profitable and not pay fees, they need to stack coins on a given account and move them from one account to another to generate large transactions without exceeding the transaction limit. Let us look at what happens if a pool of nodes is malicious towards other participants. Malicious nodes can wait for the moment when they have all the pool coins on their public key and therefore retain them. Nodes of the pool cannot prevent this kind of behaviour, which creates a high-risk environment. Pools of nodes are still possible, but trust is the essential factor in their viability, and so it is hard to create a large pool.

5.4 Fault tolerance from masternodes

Let us consider faulty masternodes that can disconnect or suffer from network latency. In PoU, when a masternode receives a block with a hash that does not match the hash of its last block (not up-to-date due to a disconnection), the masternode executes a recovery protocol. Indeed, the masternode sends requests to all other masternodes for the hashes of their last blocks and updates using the relative majority. Since a relative majority of nodes need to agree on the last block hash, the system can be considered safe. Once back on the network, faulty masternodes can retrieve all missing blocks from any proven up-to-date masternode from the majority pool.

6 Conclusion

This paper describes a new consensus algorithm that encourages participants to spend coins and share data for service provision rather than stacking. It is important to promote and sustain the usage of a stable currency over time in an environment that is both permissioned and distributed.

In its first version, Proof of Usage is open-source and accessible.[4] Currently, there are two iterations of the PikcioChain network running PoU:[5]:

- The test net, used for tests, with nine masternodes entirely governed by Pikcio.
- The main net, where any nodes can buy, sell, or exchange data. It includes 16 masternodes governed by 16 different entities, either private individuals or companies.

The changeState function of the two blockchains runs at regular intervals of 20 seconds. If the selected masternode does not share a valid block after two executions (i.e., 40 seconds), the function will consider the masternode

[4] `https://github.com/Pikciochain/Proof_of_Usage`

[5] `https://monitor.pikciochain.com/`

unavailable. At best, the test net achieves a little under a hundred transactions per second. The main net does not achieve as many transactions because the demand is insufficient. For more details, the entire test net and its transactions are available.[6]

From our point of view, the model described in Section 2 is a viable alternative to current data-exchange models, and PoU is a solution to create a reliable and scalable environment using a user-centric data-sharing model.

The next step will be to adapt the PoU consensus for public blockchains. Masternodes need to be rethought to guarantee the integrity of the blockchain even in the event that one or more masternode is malicious.

References

1. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system (2008)
2. Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, in Big Data (BigData Congress), 2017 IEEE International Congress on (IEEE, 2017), pp. 557–564
3. C. Dwork, M. Naor, in Annual International Cryptology Conference (Springer, 1992), pp. 139–147
4. M. Jakobsson, A. Juels, in Secure Information Networks (Springer, 1999), pp. 258–272
5. S. King, S. Nadal, self-published paper (2012)
6. I. Bentov, C. Lee, A. Mizrahi, M. Rosenfeld, ACM SIGMETRICS Performance Evaluation Review (2014)
7. N. company, Nem, technical reference, version 1.2.1. Tech. rep., NEM company (2018). *https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf*
8. V. Buterin, et al., white paper (2014)
9. E.B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, in 2014 IEEE Symposium on Security and Privacy (SP) (IEEE, 2014), pp. 459–474
10. I. Miers, C. Garman, M. Green, A.D. Rubin, in Security and Privacy (SP), 2013 IEEE Symposium on (IEEE, 2013), pp. 397–411
11. C. Cachin, in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, vol. 310 (2016), vol. 310
12. J. Lartigau, F. Bucamp, D.C. de Casaubon. Pikciochain: a new eco-system for personal data (2018)
13. P. Company. Pikciochain, the personal data chain, white paper version 2.0 (2018). *https://www.pikcio.com/pikciochain-whitepaper*
14. G.D.P. Regulation, Official Journal of the European Union (OJ) (2016). *https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L:2016:119:FULL*
15. M. Gill, G. Taylor, British Journal of Criminology **44**(4), 582 (2004)
16. W.H. Muller, C.H. Kalin, J.G. Goldsworth, Anti-Money Laundering: international law and practice (John Wiley & Sons, 2007)
17. R.C. Merkle, in Conference on the theory and application of cryptographic techniques (Springer, 1987), pp. 369–378
18. Q. Wang, T. Dunlap, Y. Cho, G. Qu, (2017), pp. 1–6. DOI 10.1109/WOCC.2017.7928974

[6] `https://explorer.testnet.pikciochain.com/#/`