2018 IEEE 11th International Conference on Cloud Computing

CloudPoS: A Proof-of-Stake Consensus Design for Blockchain Integrated Cloud

Deepak K. Tosh[1], Sachin Shetty[2], Peter Foytik[2], Charles A. Kamhoua[3], Laurent Njilla[4]
[1]Department of Computer Science, Norfolk State University, Norfolk, VA
[2]Virginia Modeling Analysis and Simulation Center, Old Dominion University, Norfolk, VA
[3]Army Research Lab, Adephi, MD
[4]Cyber Assurance Branch, Air Force Research Laboratory, Rome, NY
dktosh@nsu.edu, sshetty@odu.edu, pfoytik@odu.edu, charles.a.kamhoua.civ@mail.mil, laurent.njilla@us.af.mil

***Abstract*—Maintaining data provenance in cloud in a tamper-resistant manner that cannot be breached by malicious parties is a necessity from the current security standpoint. Blockchain technology has emerged as a secure solution to store and share information by offering an immutable distributed ledger service. Its effectiveness hinges on the infrastructure supporting the distributed ledger and consensus protocol that governs the validity of entries in the Blockchain. Hence, Blockchain can be a potential candidate to implement data provenance; however, traditional cryptocurrency-based consensus models become a bottleneck in the cloud environment. Therefore, in this paper, we propose a Blockchain based data provenance architecture (BlockCloud) that incorporates a proof-of-stake (PoS)-based consensus protocol (CloudPoS) for securely recording the data operations occurring in cloud environment. The critical operational phases of the protocol are discussed in depth, which leverages the cloud users' cyber infrastructure resources. A cloud-based testbed environment is created using a local cluster of physical machines managed by Xen hypervisor. Resource elasticity is enabled using Kubernetes setup that interacts with the dockerized containers, which emulate as peers in the Blockchain network. We then evaluate the effectiveness of the protocol in a simulated environment and conduct performance tests of the proposed consensus.**

***Keywords*—Blockchain, Distributed consensus, Proof-of-stake, Proof-of-work, Data provenance, Cloud computing**

I. Introduction

Assured data provenance [1] in cloud computing is crucial to keep track of data generated by operations conducted in the cloud. The state-of-the-art data provenance technologies for cloud computing involve comparing logged data generated by execution of software on physical or virtual resources with auditing data. These technologies are limited in their ability to detect integrity violations and are typically conducted in a private setting to allow better ownership of assets. However, this process is not scalable in cloud environments and at the same time it is cost prohibitive and lacks transparency. There is a need to maintain data provenance in cloud environments, where multiple stakeholders across several boundaries of trust can record the data transactions in a transparent and tamper-resistant manner. Blockchain technology offers a robust and immutable public ledger for recording transactions of any kind, and can be a core component for addressing the data provenance issue of cloud environments. Since cloud service provider (CSP) is a crucial part of the existing cloud infrastructure, it has majority control over the cloud resources. However, it is obliged to offer high quality security service to its customers. Thus, enabling Blockchain in cloud can offer Blockchain-as-a-Service (BaaS) [2][3] at a reduced cost when the underlying cloud users in conjunction with CSP can safeguard the Blockchain, where no single entity can control the Blockchain alone.

The critical component of Blockchain is the underlying consensus mechanism, which ensures a common agreement on the ledger's state and resiliency of the ledger. The objective of consensus protocols is to validate the globally accepted set of transactions, as well as a total or partial order on these transactions. The well-known proof-of-work (PoW) [4] consensus, which is widely adopted in cryptocurrencies like bitcoin and Ethereum [5], suffers from large consensus delays and high computational requirements. The Microsoft Azure's BaaS platform [2] inherently uses PoW consensus, which has scalability constraints. The well-known Practical Byzantine Fault Tolerance (PBFT) [6] consensus algorithm has been also used in permissioned Blockchain platforms like Hyperledger [7]. On the other hand, the proof-of-Stake (PoS) consensus [8] protocol takes a different approach, where the block inclusion decision-making power is given to those entities who have stakes in the system irrespective of Blockchain's length or history of the public ledger. A participant's ability to add its block in the Blockchain depends proportionately on the amount of stake it has in the system, where stake can be defined as the total currency a participant holds in the system.

However, current PoS protocols are only effective in cryptocurrency domains and cannot be effectively used in cloud computing environment. As sharing of unused resources is envisioned to be a priority in the cloud, Blockchain can come in handy dispersing the security responsibilities to every tenant in the distributed cloud environment. Therefore, resource sharing can be leveraged toward building a consensus mechanism for implementing Blockchain-based data provenance. The primary issue while designing the PoS mechanism for cloud computing platforms is defining the stake component of participants, which can be used for electing the leader who will commit the transactions to Blockchain. At the same time, the leader election process must have some entropy, otherwise the selection will be deterministic, which can be easily exploited by adversaries. The other challenge is to define the appropriate role for the CSP in maintaining Blockchain because of its involvement in performing operations, such as user authentication, resource allocation and verification, etc.

Approved for Public Release. Distribution Unlimited: 88ABW-2017-4820. Dated 28 Sep 2017

2159-6190/18/$31.00 ©2018 IEEE
DOI 10.1109/CLOUD.2018.00045

To offer a tamper-resistant, open-access, and auditable ledger for securely recording data transactions in the BaaS cloud environment, we present a Blockchain enabled data provenance model, namely BlockCloud, and propose a PoS-based consensus protocol (CloudPoS) for the BlockCloud platform. The Blockchain used in our model is considered to operate privately in the cloud environment, where the cloud users play the role of peers in Blockchain network. As the provenance feature offers additional security for the CSPs in assuring the legitimacy of data operations, cloud users can observe this as a benefit, which will motivate them to participate in the Blockchain consensus process. We consider that the Blockchain-based provenance framework will help in offering security-as-a-service from the CSP's point of view, where CSP cannot solely control the data operations in BaaS cloud environment. Instead, the CSP only plays its assigned role of the resource manager and verifier for the Blockchain consensus. By leveraging the cloud users' computing and networking resources, we formalize a proof-of-stake consensus protocol to achieve Blockchain consistency. We present an incentive mechanism to motivate the users to continue their participation in the consensus, which can be easily incorporated in the service level agreement between cloud user and service providers. To implement the CloudPoS, we have set up a testbed by creating a cloud environment on a local cluster of physical machines managed by a Xen Hypervisor. Kubernetes setup provides the elasticity of resource management that interacts with Xen Hypervisor and Docker for containerized services for communicating other peers in the network and dynamic staking. The Xen hypervisor is used to host the Kubernetes nodes in the cloud-based system. and docker is used by Kubernetes to start and stop services based on allowed resources and demand. This allows us to fabricate a distributed computing environment, where resource sharing is feasible.

This paper is organized as follows. Related works are presented in Section II. The Blockchain-based provenance architecture is discussed in Section III. In Section IV, the system model of CloudPoS consensus is described, and Section V presents the detailed phases. We present the implementation testbed and performance evaluations of our CloudPoS consensus model in Section VI. Section VII concludes the paper.

II. Related Works

As Blockchain technology is aggressively rising, there is a strong need for secure and robust consensus protocol that can avoid the double spending problem while achieving real-time performance. PoW-based [4] consensus is the core of Bitcoin that requires peers to use their computational power to change the state of Blockchain by adding a new block of transactions. In order to avoid the performance constraints, Ethereum [5] came up with a high-performing and ASIC-resistant PoW that uses memory hardness technique to demoralize orphaned block generations. However, both versions suffer from scalability issue and are vulnerable to multiple attacks [9][10]. Since the PoW consensus turns out to be energy inefficient [11], using a user's stake or ownership of currencies in the cryptocurrency system, Proof-of-Stake (PoS) [8] consensus was proposed. Proof of activity [12] uses a hybrid approach that combines both PoW and PoS to avoid security implications of both standalone models. Intel's Proof-of-Elapsed-Time (PoET) consensus [13] exploits the trusted execution environment provided

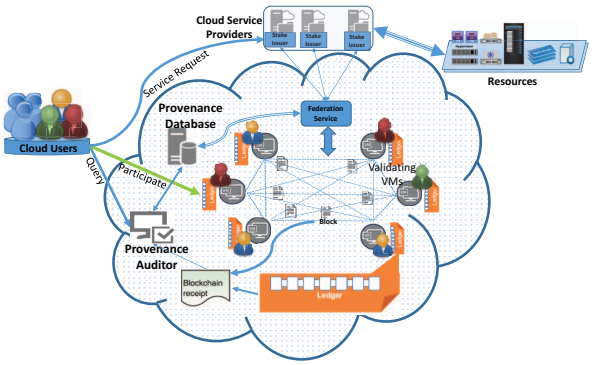

Fig. 1: BlockCloud Data Provenance Architecture

by Intel SGX processors using secure CPU instructions. With this, a guaranteed safety and random leader selection process is achieved, which does not require any powerful or specialized hardware as compared to the PoW consensus models. A novel Proof-of-Space consensus mechanism [14] exploits the disk space of users in the decentralized network that avoids the nothing-at-stake [15] problem. It uses a special type of directed acyclic graph, namely hard-to-pebble graph [16], to generate space, and the prover must store the graph in its local machine so that the verification process can be efficient.

The issues of bandwidth limitations and scalability of existing consensus models were addressed in [17] to develop a computationally-scalable Byzantine consensus protocol that has reduced message complexity. They achieve a fine-tuned performance by parallelizing the network into a number of committees that run their own authenticated Byzantine agreement protocol to agree on a value. Amalgamation of all the collected values is done at the end of an *epoch* by a designated final committee. Many of the consensus models were proposed by keeping distributed systems in mind, which may not be applicable in a Blockchain enabled cloud computing environment. In cloud, well-diverse virtualized infrastructure allows sharing of resources with third-party vendors/users, and thus, security of such platform is paramount. Since rational users will be interested in making profits by utilizing their unused resources, staking of such resources toward Blockchain consensus can be feasible in practice and has economic advantages. Therefore, revisiting the PoS consensus is important in this context instead of relying on token-based consensus, which may not be directly applicable in the cloud environment.

III. BlockCloud Architecture

Our system model consists of the typical cloud computing paradigm, independent of the application it is used for, where each cloud environment hosts one or more virtual resources (virtual machines [VM]). Each VM is owned by a cloud user and comprises an operating system, software, data, etc. An executing VM creates dynamic data, which is key for provenance. As such, and in cloud computing today, provenance is provided on the cloud through the linking of log data (data that is generated through the execution of software on the given physical, virtual, or application resource) and audit data (data that is created for the sole purpose of provenance assurance).

The Blockchain technology will be beneficial to cloud services that have a strong attribution for supporting assured data provenance in the cloud domain [18]. To enable data integrity

over the public ledger in a Blockchain cloud, cryptographically enforced blocks join in the Blockchain after a consensus is reached in the decentralized network, where transactions in the blocks are authenticated by peers of the network. This shared ledger could potentially contain a history of every transaction on the data objects owned by different cloud users, and the authenticity could be verified without involving a cloud administrator. The combination of a cryptographic mechanism and a decentralized public ledger helps in building applications on top of the Blockchain without worrying about the trust components of users and maliciousness in the Blockchain-enabled cloud system.

BlockCloud is our proposed data provenance architecture built using a Blockchain that provides the ability to audit data-related operations for cloud users and providers in the cloud. The BlockCloud architecture achieves the following four goals.

1) *Cloud Data Provenance:* User operations are monitored in real time to collect provenance data to support access control policy enforcement [19] and intrusion detection.
2) *PoS Validation:* The consensus process in the Blockchain network are validated by trusted VMs housed in a cloud computing environment. The presence of validating VMs will provide supervisory control over the consensus process.
3) *Tamper-proof Environment:* Data provenance records are collected and then published to the Blockchain network, which protects the provenance data. All data on the Blockchain is shared among Blockchain network nodes. BlockCloud builds a public time-stamped log of all user operations on cloud data without the need for a trusted third party. Every provenance entry is assigned a Blockchain receipt for future validation.
4) *Provenance Data Validation:* A data provenance record is published globally on the Blockchain network, where a number of Blockchain nodes provide confirmation for every block and using Blockchain receipts every provenance information can be validated.

Our proposed BlockCloud architecture, as depicted in Figure 1, achieves the above objectives by monitoring user activities in real time using hooks and listeners (special classes of event listeners) so that every user operation on files will be collected and recorded for generating provenance data. Each piece of provenance information is referred to as transactions that are broadcasted to the core of the Blockchain network created by a specific set of validating VMs. The validators collect the raw transactions and create their blocks individually, then wait for the consensus to converge, after which a leader will be selected to extend the Blockchain. Every validator node on the Blockchain network can verify the data operations, or transactions before including them into the block to ensure that the provenance data is authentic. We consider the presence of a provenance auditor in our model, who collects the confirmation of each transaction block that is successfully added into the Blockchain and records it in the provenance database.

IV. Consensus Model Description

In our model, the consensus execution occurs in distinct time slots, namely epochs. At the end of every epoch, the outcome is to successfully commit a block of transactions, thereby expanding the Blockchain one block at a time. During every epoch, several other tasks, such as leader election, transaction validation, and multiparty signature, are executed before committing the block. In our system, we consider that any cloud user can freely participate in the Blockchain consensus process given it can verify its identity and affiliation with CSPs. Without loss of generality, the goal of the protocol is to reach a consensus and agree on the latest state of the distributed ledger containing cloud data transactions.

We consider that N fully connected cloud nodes $\{P_1, P_2, \cdots, P_N\}$, also referred to as validators, participate in the consensus process. The peers in this context are the virtual nodes in the cloud computing architecture, and the N nodes are responsible for maintaining the Blockchain in full. The other peers or users of the cloud system, who manipulate various data objects, inherently create transactions that are broadcasted to the P2P network of N validators. Before the consensus mechanism kicks off, the validators selectively gather transactions without disturbing the order of data object manipulations in order to create candidate blocks. We also assume that there exists a limit on the number of transactions to be included in a block, otherwise it may cause overhead in the network while replicating at other peers. In the cloud computing architecture, we consider the cloud service provider (CSP) as a part of the system, who may act the role of membership manager [20] in the provenance system. Similar to the case of cryptocurrency, where members must have some initial balance in order to perform transactions. We also consider that cloud validators must have preoccupied virtual resources from CSP, which will be used as virtual currency in the consensus.

A. Threat Model

As users in the cloud computing environment are agile and volatile in nature, the validator nodes may enter and exit the consensus process anytime. Any new validator may be allowed to enter the system only at the end of an epoch so that the running consensus slot will be unperturbed. However, some validators may behave erratically or even disappear in between an ongoing epoch of the consensus, thus the leader's block may be discarded and a new leader will be re-elected in this situation. Permitting any cloud user to be a validator and flexible rules of entry/exit to/from the consensus system can expose the attack surface for maliciousness, where malicious nodes can enter as validators and try to be a leader by staking the most resources in order to include their own transactions in the Blockchain. To avoid such maliciousness, the CSP keeps a record of every validator, and the list is updated at the end of each epoch. Furthermore, this auditing will also assist in building reputation of the validators over time to dynamically derive the incentives based on their reliability and reputation. The incentive of validators has a correlation with how truthfully they serve for the consensus and their implicit greediness factor (γ), which is a decisive parameter for the validators to decide how much resource to stake at each epoch.

B. Stake in Cloud Computing

Resources in cryptocurrency domain are nothing but the tokenized form for the currencies, which are used for staking in the PoS consensus. However, in the context of cloud computing, we categorize the resources in the form of (1) CPU power or the number of CPU slices/cores provided by the CSP, (2) Amount of memory allocated for program execution and temporary buffer, and (3) Network capability to communicate

with other peers, which can also refer to the network data rate. In general we consider a user $i \in N$ is initially allocated with a total cumulative resource of $R_i =< C_i^{max}, S_i^{max}, D_i^{max} >$, where C_i^{max} is the number of CPU slices, S_i^{max} is the size of memory allocated in kilobytes, along with the network data rate of D_i^{max} Kbps. A portion of these resources is generally used for staking purposes, and the amount to stake is independently decided by the users, depending on their greediness parameter. The following Section describes the phases of consensus and the overall mechanics in the detail.

V. The CloudPoS Consensus

A. Overview of CloudPoS

The CloudPoS consensus initially requires the validators to decide what amount of resources they want to stake toward the consensus process, which is driven by the fact that chances of becoming a leader in the epoch increases as the amount of resources staked are increased. Without loss of generality, we model a stake function $f(.)$ for a validator i that is dependent on its total allocated resource R_i, implicit greediness parameter γ_i, and current resource utilization vector (R_i^u). The function results in the stake vector $\mathcal{X}_i =< \mathcal{X}_{C_i}, \mathcal{X}_{S_i}, \mathcal{X}_{D_i} >$ for validator i, based on which this amount of resources is sliced off from the total allocated resources. Then the leader gets elected stochastically based the individual stakes $\bar{\mathcal{X}} = \{\mathcal{X}_i : i = 1, \cdots N\}$. If the leader is unavailable/unreachable, then reelection for a new leader occurs. After a leader is successfully selected, its block is propagated to perform a consistency-check on the transactions and match the Merkle root of the block to ensure that the leader's working chain aligns with other validators' chains. Finally, the leader collects its reward that was allocated for that particular epoch. Our CloudPoS consensus is executed in several phases during a particular epoch to finalize a block of transactions to include in the mainstream Blockchain. Afterwards, the same steps are iterated after the length of Blockchain increments by one block. The five phases that constitute one epoch of the consensus are (1) Stake determination, (2) Resource staking and confirmation, (3) Leader election, (4) Block replication and verification, and (5) Reward distribution, which we briefly describe in the following subsections.

B. Stake Determination

The idea of resource staking comes from the concept of coin staking in cryptocurrency domain [8][12]. However, determining the amount of stake to make, being a validator in the consensus, clearly depends on its risk-taking behavior and how often the validator requires use of the staked resources. In the sense of cloud computing domain, the decision of stake amount is required to be taken strategically by considering current resource usage and future demand.

Now, from the perspective of a single validator i, it is known that the total resource allocation tuple for i is $R_i =< C_i^{max}, S_i^{max}, D_i^{max} >$, where C_i^{max}, out of which it needs a specific amount R_i^u to run its normal business operations. Let us define this resource demand as tuple, $R_i^u =< \tilde{C}_i, \tilde{S}_i, \tilde{D}_i >$. We considered that each validator i has an implicit greediness parameter $\gamma_i \in (0, 1]$, where the higher the value of γ, the greedier the validator is toward staking its available resources for consensus. Introducing such a parameter allows us to bring heterogeneity in the system, and therefore validators are not likely to stake an equal amount of resources as stake. On a side note, we can also break the greediness component into three different components, where each component can represent the priority for the individual piece of resource (from CPU power, memory, and network). In that way, the overall greediness factor can be a weighted sum of all three pieces, e.g., $\gamma_i = w_1\gamma_{cpu}^i + w_2\gamma_{mem}^i + w_3\gamma_{nw}^i$, where $\sum_{k\in\{1,2,3\}} w_k = 1$, and w_1, w_2, w_3 are the scaling parameters. Simplistically, we can adopt the stake function as the following linear model:

$$f(R, R^u, \gamma) = \gamma(R - R_u) \tag{1}$$

Thus, for validator $i \in N$, the stake $\mathcal{X}_i$ can be represented as $< \mathcal{X}_{C_i}, \mathcal{X}_{S_i}, \mathcal{X}_{D_i} >$, where

$$\mathcal{X}_{C_i} = \gamma_{cpu}^i(C_i^{max} - \tilde{C}_i) \tag{2}$$

$$\mathcal{X}_{S_i} = \gamma_{mem}^i(S_i^{max} - \tilde{S}_i) \tag{3}$$

$$\mathcal{X}_{D_i} = \gamma_{nw}^i(D_i^{max} - \tilde{D}_i) \tag{4}$$

The above-defined stake components for the validators change over time as the consensus progresses because the utilized resource amount changes temporally. In order to calculate the stakes, an average value of total used resources at the beginning of an epoch is taken to consideration and kept fixed throughout the epoch.

C. Resource Staking and Confirmation

After estimating the amount of resources the validators decide to stake, in this phase they are required to lock those resources, which we meant as staking. By doing so, they cannot access the staked resources until they decide to exit the consensus system and/or change their validator role. The requirement of locking the resource is stringent because of the possibility of double spending attack in cloud computing. It is important for the validators to prove that they have staked their resources to participate in the system, where those staked resources are out of their hands. There are two ways to achieve this task. One way to achieve that is by involving CSP as a resource controlling entity during the consensus, where the stake amount $\mathcal{X}_i$ from each validator's system is taken away temporarily by the CSP, which can later be provisioned back when the node i decides to not participate as a validator. However, involvement of CSP destroys the decentralized characteristics of the consensus system, therefore it is not a preferable option.

The other option for a validator i is to solely instantiate a VM that consumes $\mathcal{X}_{C_i}$ CPU slices, $\mathcal{X}_{S_i}$ amount of memory, and $\mathcal{X}_{D_i}$ amount of networking bandwidth from the available resources. In order to restrict the access to the newly instantiated VM, a shared secret will be generated and we can use threshold encryption technique [21][22] for encrypting the secret. However, the decryption of the shared secret requires collaboration of a predefined number of validators as well as the CSP. We intentionally include CSP for securely locking the stakes because involvement of only validators in access restriction may open opportunity for collusion, where multiple malicious validators can join the system to bypass this issue. For resource staking, we use the following primitive:

$$\texttt{VMCREATE}(< \mathcal{X}_{C_i}, \mathcal{X}_{S_i}, \mathcal{X}_{D_i} >, SS) \rightarrow (\Delta_i, txID_i), \forall i \in N$$

where, SS is the share secret that is provided by CSP while a peer joins as validator. The outputs of the VMCREATE primitive are the signature (Δ_i) of created instance at validator i and the transaction confirmation ($txID_i$). These two parameters are required to periodically check if the validators have a running instance of matching signature or not. Once the stake has been instantiated at the validator node, the corresponding transaction confirmation and signature of the instance is included in the block, so that in the later stages when a leader tries to update the Blockchain with its own block, the other validators can verify the stake instances in real time using the primitive, $\texttt{VMVERIFY}(\Delta_i) \rightarrow 0/1$, which has a binary output of 1, if instance is running, otherwise 0.

D. Leader Election

In this stage, the leader for the current round is selected, and the probability of validator i becoming a leader depends on its staked resource amount $\mathcal{X}_i$. We denote $\|\mathcal{X}_i\|$ as the magnitude of the staked resources, which can be used to compare the stakes from other validators. In this way, it will be easier to compute the chances of a validator to become the leader for the current round. The leader election with respect to the stake distribution of $\{(pk_1, \|\mathcal{X}_1\|), (pk_2, \|\mathcal{X}_2\|), \cdots, (pk_N, \|\mathcal{X}_N\|)\}$, which is known to every validator in the system, will be biased toward a validator $i \in N$ with the following probability $p_i = \frac{\|\mathcal{X}_i\|}{\sum_{k=1}^{N} \|\mathcal{X}_k\|}$.

In order to execute the leader election process, where a leader will be selected based on the probability distribution $\{p_i\}_{i=1}^{N}$, we can adopt a timer-based voting approach similar to Raft consensus [23]. However, the differentiating point is that the validators will initiate a random biased timer based on their stake proportion in the system. At the time-out event, the validator sends out a request for vote. The receiving validators perform a validity check of the request and respond with binary output. We consider that every epoch can have multiple rounds of leader selection until a single leader is selected. Once the epoch is changed, the value of the round resets back to 0. In the phase of leader election, we assume that the validators communicate using remote procedural calls to vote candidate leaders. Every request includes the information of epoch and round, so that the validators can decide whether the request is stale or fresh, and respond accordingly.

After receiving a request, the validator first stops its own timer and performs a validity check to decide if the current epoch and round information match with the corresponding information in the request. After that, the $\texttt{VMVERIFY}(\Delta_i)$ construct is invoked to ensure that the stake is actually instantiated. If every condition is satisfied, the validator can respond with a positive vote to the sender, otherwise it drops the request. The validator receiving a majority of the votes eventually wins the election and proceeds to the next phase. However, there might be chances of re-election, where the election ends up having no winner at all. This situation arises when multiple validators are having split majority votes, then a single winner is not derivable. The election timers are biased and randomly sampled from a fixed interval. Ideally, the average value of time out for a validator with high stake will be less than a validator with low stake in the system. With such property, validators' chances of winning the leader election are fairly distributed according to their stake value.

Algorithm 1: $\texttt{SelectLeader}(\{\mathcal{X}_k : k \in N\})$ Leader Election procedure at validator i at *epoch* t

Input: *Epoch* (t), Probability distribution of stakes ($\{p_1(t), p_2(t), \cdots, p_N(t)\}$), $maxTimeout$

$rTime \leftarrow Uniform\_Random()$;
while $rTime < p_i(t)$ **do**
  Start timer for $maxTimeout \times (1 - p_i(t))$ sec;
  if *received a leader voting request from validator* k *and the timer is still running* **then**
    Stop the timer and acknowledge (ACK) the leader if request's *epoch* matches;
    Count the number of ACKs to ensure majority;
    if *Majority ACKs received* **then**
      Select k as candidate leader;
      $status_k \leftarrow VMVERIFY(\Delta_k)$;
      if $status_k = 1$ **then**
        Select k as leader;
        Send confirmation to validator k;
        exit while loop;
      else
        Restart leader election from step 1;
      end
    end
  else
    If time-out occurs then break, else go to step 4;
  end
end
if *time-out occurred and leader not found* **then**
  Broadcast leader voting request to all;
  Wait for $mTimeout$ sec to recieve ACKs;
  if *# of ACKs received*$> \lceil \frac{N}{2} \rceil$ **then**
    Select validator i as leader;
  else
    Restart leader election from step 1;
  end
end

Algorithm 1 outlines the leader selection process, which works on the given stake proportions of all validators at epoch t. We use two separate messages (request and acknowledgment) for communicating between validators in the phase of leader election. As the stake proportion vector $\mathcal{P} = \{p_1, p_2, \cdots, p_N\}$ is nothing but a probability distribution for leader selection, the i^{th} component refers to the power of validator i in the system. With probability p_i, the validator i starts the leader election process, where the validator initializes a stake-biased randomized timer. The probability distribution of stakes ensures that the random time-out period is distributed proportionately based on the staked amounts. Given the validators need wait for at max of $maxTimeout$ time units before they can send the voting request, the actual waiting period of a validator might be less depending on its total stake allocation amount. If i has higher stake, the probability of becoming a leader (p_i) is high and thus the waiting period ($= maxTimeout \times (1 - p_i)$) is small. This stake-biased timer ensures the fairness of the leader selection process by allowing a validator with high stake to be selected as leader more times compared to a validator with low stake. However, if the timer is running and a request comes from another validator, then

the timer is paused and the request is validated to ensure that it is not an older request that came late by comparing the epoch value in the request with known epoch. In case of mismatch, the validator discards the request and resumes the timer.

The validator sends a positive acknowledgment (ACK) to the request and waits for more ACKs to come from other validators in order to compute if the leader has received the majority of votes or not. In case of majority ACKs collected, the leader is chosen to be a candidate and proceeds toward stake verification. In this phase, every validator verifies whether the selected candidate leader has actually staked the reported amount of resources by querying with the $\texttt{VMVERIFY}(\Delta_i)$ construct. If the staked resource is active, the former primitive will return true, after which the validators can send confirmations to notify the candidate as the selected leader for the epoch. On the other side, if the running validator's timer times out before it receives any external requests, it first tries to send voting requests to other validators in the network and waits for ACKs. In case of majority ACKs, the validator itself is selected as a candidate leader. If the stake is properly allocated and successfully verified by the peer validators, the confirmation messages will follow to notify that it has been elected as leader.

E. Block Replication and Verification

To achieve election safety, the above phase ensures only a single leader is elected for the current epoch. The elected leader has the right to include its own block of transactions given all other validators agree on this fact. It is assumed that the validators create their transaction block at the very beginning of an epoch by preserving the order of transactions. The data structure of a block includes a header part, and the rest is a list of transactions. In our case, the block header contains extra information regarding the stake of the leader, such as $\Delta_i, txID_i$. The other attributes, such as the previous block's hash, timestamp, Merkle root of transactions, and block height (epoch) are kept unaltered. The leader's block is sent to all the remaining validators, who will append the block to their Blockchain after verification of the candidate block.

The transactions included in the block must be verified by the validators to ensure their order of manipulation of cloud data objects is consistent according to their observed transaction list and transactions committed in the previous block. This check is mandatory for the validators because it can help preventing double spending. Furthermore, the consistency check can be made stricter by asking for confirmations from other validators regarding the transaction ordering. However, this may introduce extra communication overhead as well as increase the duration of an epoch. The integrity check is another important validation criterion to ensure that the transactions are not altered during the replication.

F. Incentivization

The sanctity of the CloudPoS consensus solely depends on the active involvement of validators, who are responsible for performing major tasks such as transaction gathering, and Blockchain extension by staking their useful resources to the system. It is necessary to offer economic incentives to the volunteering validators in return for their service in maintaining the Blockchain. In a cryptocurrency case, rewarding miners was easier to achieve, where the miners create a default transaction that rewards a certain amount of coins to one of their owned address. However, it is challenging to define reward for the virtual validators in the cloud computing scenario. There can be several ways to incentivize the validators. One way is to involve CSP who will reduce the cost of resource leasing by a certain percentage for the winning validator for a predefined number of epochs. By doing so, the overall cost of leasing resources including the staked resources will be less compared to when the node does not participate as a validator. Another way to incentivize a validator upon successfully extending the Blockchain can be done by releasing some of the staked resources back to the validator without decreasing the value of its stake. In this way, the validator can increase its workable resources by truthfully participating in the consensus.

The other crucial factor to determine is how much incentive to offer when the validator correctly includes its block in the Blockchain. Although it is intuitive to say that the reward must follow the distribution of staked resources, quantitative analysis is needed to understand what reward amount can be sustainable from a CSP's point of view. The CSP must evaluate the total value of providing provenance service to the validators as well as other clients for a specific period of time, based on which cost-benefit analysis can be conducted to determine the total reward value (R_{total}) for the validators. Then, the total reward can be distributed depending on the winning validators' stakes over the considered cycle for incentivizing them.

A simplistic reward function for a particular validator i can be derived by considering various parameters such as: the amount of staked resources, the number of validators participating, the duration of staking, and the number of times selected as leader. These parameters describe how actively the validator is involved in the consensus process, and therefore the amount of reward to distribute must follow accordingly. Assuming the provider has allocated a total amount of reward R_{total} for incentivizing the validators over the duration of Z epochs, we aim to find the reward distribution $\mathcal{R}_Z = \{R_{t+1}, \cdots, R_{t+Z}\}$ such that $\sum_{i=1}^{Z} R_{t+i} = R_{total}$. The reward of one epoch, say $t+t'$, not only depends on the leader's stake proportion but also on the other participating validators who support the block/stake validation process. Although most of the single epoch reward goes to the leader, it is necessary to incentivize other validators for their service too. Otherwise, all the validators except the leader will have no motivation to cooperate in performing any transaction required by the consensus. A simple way to distribute rewards can be done by allocating an equal amount of incentives to each epoch, i.e., $R_{t+k} = \frac{R_{total}}{Z}, \forall k = 1, \cdots Z$. Then, the slot k's reward can be redistributed fairly depending on the leader's stake amount and other validator's involvement.

VI. Experimental Setup and Evaluation

A. Testbed Environment

The cloud-based testbed environment chosen is based on a local cluster of physical machines managed by a Xen Hypervisor. Elasticity resource management software like Kubernetes is used to interact with the Xen Hypervisor and Docker is used for containerized services. The actual implementation of the Blockchain system is built with a combination of python scripting and C# applications. The Xen Hypervisor is used

to host the kubernetes nodes in the cloud based system, and Docker is used by kubernetes to start and stop services based on allowed resources and demand. The docker containers are created ahead of time and include the validator and controller containers. The validator container offers a P2P service between all participating validators assigned in the network. The controller acts as a bridge point in which transactions can originate and membership services can be handled. Docker allows images of the container to be created dynamically and these images can be put on any machine that would potentially start up and provide the service of that container.

Kubernetes requires a node in the cloud to be set up as controller or master node and then all other participating nodes need to have a kubernetes node controller running on them. The controller is configured to direct nodes to spin up the assigned applications and to use a defined amount of resources. If a docker container goes down on a particular node, kubernetes will have redundant containers already running to ensure availability of the services. A validator node (a container service running on a kubernetes node) allocates the resources as stake within the system. It communicates the stake information to the Kubernetes controller, which allocates the resources by running a test application that consumes the resources according the staked amount. These consumed resources can be queried for checking by other peer nodes.

B. Test Environment

By implementing the application within a simulated environment first, a full spectrum of tests are performed on the proposed algorithm. This allows for testing of extremes as well as provides better insights to optimize the performance of our testbed. The simulation is developed as a test application, where a number of validator nodes, a simulation controller, and associated Blockchain infrastructure are spun up when started. The validator object has the ability to choose a desired stake based on a random distribution or a set of values for testing. The desired stake value is used as the mean value in a normal distribution that is used to select a validator's current stake. Each validator's stake value is chosen between 0 and 100, which would be the amount of resources the validator is putting on hold for the PoS system. A delay is introduced to represent time taken for successful resource staking, and another delay is to represent network congestion. The performance metrics used to evaluate the CloudPoS are (1) total number of times a leader was selected as validator but did not have the highest stake amount; and (2) average, max/min time in milliseconds to make progress and extend the Blockchain with a new block in absence as well as presence of network delay. These performance metrics were recorded by varying the number of validator nodes and transactions for 30 different tests to allow for a good representation of the results, since the simulation uses stochastic variables for selection of stake and delay.

Figure 2 illustrates the performance of the PoS-enabled Blockchain given an ideal scenario of negligible network delay and stake allocation delay. In this case, the latency of the consensus process is dependent on the leader selection procedure. This latency variation over 30 experiments gives us an estimated ideal time of extending the Blockchain with a new block. In other words, the duration of an ideal epoch is going to be in the range of 0.4 to 0.8 milliseconds when there

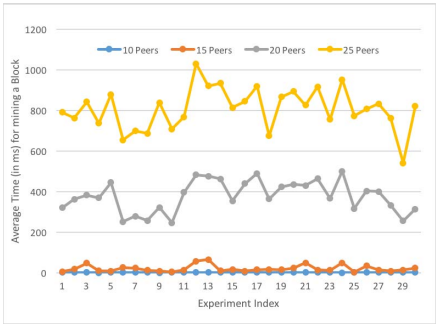

Fig. 2: Average time to extend Blockchain with a new block

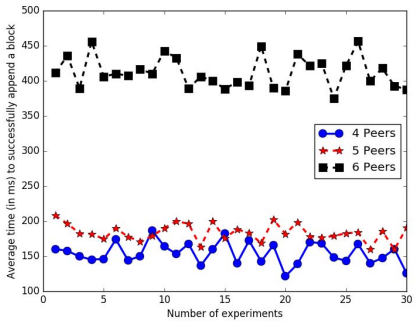

Fig. 3: Average time (in ms) to extend Blockchain with a new block in presence of network delay

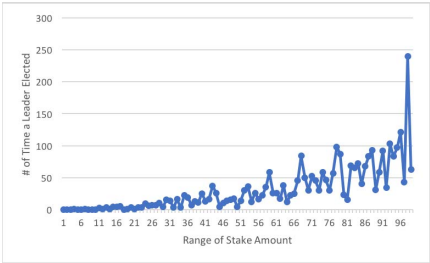

Fig. 4: Average number of times a leader elected ordered on amount of its stake

are 10 validators and 5 to 10 milliseconds when 15 validators are present. As the number of validators increases, it is obvious that the leader election process takes more time to definitively find a leader, which is why average latency is increased. Thus, it is important to decide a threshold of how many validators to lead the Blockchain consensus process. In Figure 3 a similar experiment is conducted but with the introduction of random communication delay. The delay component is sampled from a normal distribution in the range of 1 to 5 milliseconds. The 30 experimental rounds show that even with the presence of network delay, our PoS consensus eventually extends the Blockchain and hence satisfies the safety and liveness properties.

We also experimented to verify that the majority stakeholders do not always become leader, rather the minority group of validators do get chance to extend the Blockchain by becoming a leader. For this, we tested our simulated prototype for 100 blocks and checked how many times a leader is selected in an epoch that does not have highest stake in the system. The evaluation depicted in Figure 4 shows the leader distribution according to the stake values ordered in increasing fashion. It is evident that a validator from the minority stakeholders

is selected more than 50% of the times on average, which means that the highest stake holding validator does not get elected as leader all the time. It is interesting to notice that as the number of validators increases, the selected leader is most likely to fall in the minority category because the stake values will be closely distributed and hence the election time-outs will also be similarly distributed. Thus, it is preferred to have more validators in the system to make the leader selection process more fair.

Optimization Goals: Current implementation considers that the stake allocation and verification occur at the beginning of every epoch, which can be a major bottleneck in the protocol. Since stake allocation requires adjustment to the actual resource utilization dynamically, it could result in extra delays before any validator can verify the staked resource. In order to bypass this issue, we can enforce the validators to stake their resources for a fixed amount of time that is more than a typical epoch duration. A smart contract can be used for this purpose to prohibit release of stakes before the agreed staking time, which can be a regulatory condition for becoming a validator in the distributed computing system.

VII. Conclusions

This paper presents a Blockchain-based data provenance framework for BaaS cloud computing environment, where the Blockchain is driven by a novel proof-of-stake consensus model. Staking the cloud computing resources, the users can gain the privilege of becoming validators in the consensus process and the motivation of such participation can be either rewards, or achieving high security through collaborative provenance. The proposed leader election process ensures that every cloud user gets a fair chance of leading the Blockchain based on the amount of resources it staked and thus the maliciousness in the Blockchain environment is reduced because the resources of participants are at stake. We have developed a sandboxed environment on the Kubernetes platform to evaluate CloudPoS and demonstrated the effectiveness of the protocol by conducting performance evaluation through simulation in presence of network latency. Although the cloud service provider has an important role in the current framework, it cannot be eliminated unless a fully decentralized cloud is introduced. In future, we plan to study the PoS consensus under minimal involvement of CSP.

Acknowledgment

This material is based on upon work supported by the Department of Energy under Award # DE-OE0000780, Air Force Material Command under Award # FA8750-16-0301 and Office of the Assistant Secretary of Defense for Research and Engineering (OASD (R&E)) agreement FA8750-15-2-0120.

Disclaimer

References

[1] B. Lee, A. Awad, M. Awad, Towards secure provenance in the cloud: a survey, in: Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on, IEEE, 2015, pp. 577–582.

[2] Microsoft azure blockchain as a service, https://azuremarketplace.microsoft.com/en-us/marketplace/apps/microsoft-azure-blockchain.azure-blockchain-service?tab=Overview, Last Accessed: March, 2018.

[3] Ibm blockchain solutions and services, https://www.ibm.com/blockchain/offerings.html, Last Accessed: March 6, 2018.

[4] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008).

[5] V. Buterin, et al., Ethereum white paper (2013).

[6] M. Castro, B. Liskov, et al., Practical byzantine fault tolerance, in: OSDI, Vol. 99, 1999, pp. 173–186.

[7] C. Cachin, Architecture of the hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[8] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, self-published paper, August 19, 2012.

[9] D. K. Tosh, S. Shetty, X. Liang, C. A. Kamhoua, K. A. Kwiat, L. Njilla, Security implications of blockchain cloud with analysis of block withholding attack, in: Proceedings of 17th IEEE/ACM Intl. Conf. on Cluster, Cloud and Grid Computing, CCGrid, 2017, pp. 458–467.

[10] D. K. Tosh, S. Shetty, X. Liang, C. Kamhoua, L. Njilla, Consensus protocols for blockchain-based data provenance: Challenges and opportunities, in: 8th IEEE Conf. Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017, pp. 469–474.

[11] K. J. O'Dwyer, D. Malone, Bitcoin mining and its energy footprint.

[12] I. Bentov, C. Lee, A. Mizrahi, M. Rosenfeld, Proof of activity: Extending bitcoin's proof of work via proof of stake, ACM SIGMETRICS Performance Evaluation Review 42 (3) (2014) 34–37.

[13] Sawtooth lake, https://intelledger.github.io/introduction.html.

[14] S. Dziembowski, S. Faust, V. Kolmogorov, K. Pietrzak, Proofs of space, in: Annual Cryptology Conference, Springer, 2015, pp. 585–605.

[15] A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A provably secure proof-of-stake blockchain protocol, in: Annual International Cryptology Conference, Springer, 2017, pp. 357–388.

[16] J. Alwen, V. Serbinenko, High parallel complexity graphs and memory-hard functions, in: Proceedings of the forty-seventh annual ACM symposium on Theory of computing, ACM, 2015, pp. 595–603.

[17] L. Luu, V. Narayanan, K. Baweja, C. Zheng, S. Gilbert, P. Saxena, Scp: A computationally-scalable byzantine consensus protocol for blockchains., Cryptology ePrint Archive, Report 2015/1168, 2015 http://eprint.iacr.org/.

[18] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, L. Njilla, Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability, in: International Symposium on Cluster, Cloud and Grid Computing, IEEE/ACM, 2017.

[19] J. P. D. Nguyen, R. Sandhu, Dependency path patterns as the foundation of access control in provenanceaware systems, 4th USENIX Workshop on the Theory and Practice of Provenance, TaPP'12, 2012.

[20] X. Liang, S. Shetty, D. Tosh, P. Foytik, L. Zhang, Towards a trusted and privacy preserving membership service in distributed ledger using intel software guard extensions, in: Intl. Conference on Information and Communications Security, Springer, 2018, pp. 304–310.

[21] Y. Desmedt, Threshold cryptosystems, in: International Workshop on the Theory and Application of Cryptographic Techniques, Springer, 1992, pp. 1–14.

[22] J. Baek, Y. Zheng, Identity-based threshold decryption, in: Public Key Cryptography, Vol. 2947, Springer, 2004, pp. 262–276.

[23] D. Ongaro, J. K. Ousterhout, In search of an understandable consensus algorithm., in: USENIX Annual Technical Conf., 2014, pp. 305–319.