Oct 20th, 11:55 AM - 12:20 PM

A Blockchain-based Security-Oriented Framework for Cloud Federation

Ramandeep Kaur sandhu
Virginia Commonwealth University, sandhurk2@vcu.edu

Kweku Muata A. Osei-Bryson
Virginia Commonwealth University, KMOsei@vcu.edu

Abstract
Cloud federations have been formed to share the services, prompt and support cooperation, as well as interoperability among their already deployed cloud systems. However, the creation and management of the cloud federations lead to various security issues such as confidentially, integrity and availability of the data. Despite the access control policies in place, an attacker may compromise the communication channel processing the access requests and the decisions between the access control systems and the members(users) and vice-versa. In cloud federation, the rating of the services offered by different cloud members becomes integral to providing the users with the best quality services. Hence, we propose an innovative blockchain- based framework that on the one hand permits secure communication between the members of the federation and the access control systems, while on the other hand provides the quality services to the members by considering the service constraints imposed by them.

Location
KC 462

Disciplines
Information Security | Management Information Systems | Technology and Innovation

1 INTRODUCTION

Recent trends and advances such as cloud computing have made it possible to collect, store and analyze data more pervasively, efficiently, and effectively [3]. But as more and more information gets placed on the cloud, the concerns about how safe the cloud environment is, are beginning to emerge [15], [16]. *The users of the cloud computing don't have the control over the data placement, cost and the execution time associated with the services provided in the cloud computing. Hence, quality of service issue (QoS) may arise during the cloud computing* [13] **(Issue 1)**. Since users no longer physically own the storage of their data, traditional security mechanisms cannot be applied to secure the data in cloud computing. Therefore, security and privacy issues inhibit the enterprise customers from establishing their businesses in the cloud [4]. And recently, there is a trend of cloud federations, where members of federation share data and services hosted on their platforms with the other members in the federation. Kurze et al.2011 defines cloud federation as: "*Cloud federation comprises services from different providers aggregated in a single pool supporting three basic interoperability features - resource migration, resource redundancy and combination of complementary resources resp. services*". But, many times resource sharing in cloud federation can be obstructed by access control requirements set by the resource owner. To address this issue, there has been federation wide access control systems deployed to enforce the access control policies set by the resource owner [17]. This means that there will be distributed components that will receive, process access requests and access decisions virtually. Hence, there is a possibility that exchanged access request and access decision messages may be subverted, and user's data may be accessed by other unauthorized users. [4], [7] **(Issue 2)**. Moreover, to encourage wide adoption of cloud federations, federated clouds advocates the decentralized and democratic federation governance. This means that amongst the federation members, there can't be a designed leader responsible for data governance, whereas federation members form the network of peers. An exemplar of this trend is provided by federation as a Service FaaS cloud federation [14]. Hence, there is a possibility that an attacker may violate database integrity by directly altering the database **(Issue 3)**. A federation member may modify the database without informing other members leading to integrity violation (**Issue 4**). Multiple members of cloud federation may also conspire together to alter the database malevolently [8] **(Issue 5**).

To impede these collusion attacks, *Blockchain* technology can be seen as potential candidate for designing and implementing a distributed, secure database for cloud federations. Blockchain enables a trustless distributed peer to peer network where non-trusting members can transact without a trusted intermediary. Not having a trusted intermediary means quicker reconciliation between the transacting parties [5]. Blockchain has also received considerable momentum for enthralling properties it guarantees (For instance: persistency and non-repudiation of the data and distributed consensus) [8]**.** Blockchain is comprised of consecutive blocks which are linked together in a decentralized peer to peer network. The original mining process still used in Ethereum and bitcoin consist of blocks created in a decentralized fashion through the

consensus algorithm called "Proof-of-Work". Each block consists of the transactions, which the nodes have agreed upon through the distributed consensus. Once the block becomes the part of the block chain, it becomes non-repudiable and persistent, "unless an attacker has the majority of miner's hash power to create a fork of chain" [8]**.** In the context of cloud federations, the blockchain could be used to establish the database with strong integrity guarantees. But, the performance achieved with Proof of Work based blockchain is poor, as compared to the technologies used in traditional databases. The lack of performance is due to the time-sensitive task of the *Proof of Work* and broadcasting latency of the blocks on the network. All the transactions stored on the blockchain have high confirmation latency, which leads to extremely low throughput. [5], [8] **(Issue 6**). It is also difficult to maintain *privacy* on the blockchain. Each participating node is recognized by their public keys. The participating nodes just needs to know the keys of their transacting counterparties. Given that the blockchain transactions happen in open, by analyzing this data, the attackers/ interested parties can recognize patterns and create connections between addresses and make informed inferences about real identities behind them. [5] (**Issue 7**).

Despite the intrinsic features of blockchain, the issues of poor performance and lack of privacy make the blockchain incompatible with the cloud federations. Therefore, the objective of this study is to deploy a blockchain-based cloud federation framework, with the improved performance compared to PoW based blockchain "as-is", while preserving the required guarantees on data integrity, security, privacy as well as quality services. It should be noted that several subsets of these challenges have been addressed in previous research, but no single previous study has addressed all.

CONTRIBUTION

This study builds and extends the previous work by Gaetani et al. [8] in the scope of cloud federations, which assures data integrity guarantees. The study by Gaetani et al. [7] doesn't address the other important issues such as secure commination between different components in cloud federation, privacy as well as QoS constraints imposed by cloud federation members. In this study, we extend the proposed framework by Gaetani at al. [8] for secure communication between different components in cloud federation as well as address privacy and QoS related issues of cloud federation members. To summarize, our first contribution is integrating all the issues related to data integrity, secure access controls and QoS constraints specific to cloud federation in one study. The second contribution is an innovative blockchain based framework that ensures secure communication between the members of the federation and the access control systems and privacy. The third contribution is that our proposed solution provides quality services to the members by considering QoS constraints imposed by the members of the cloud federation.

STRUCTURE OF THE PAPER

Section 2 reviews related work. Section 3 presents the proposed framework and various components in the framework. Description of the procedure for secure communication between

access control systems and federation members is depicted in Section 4. The proposed framework is evaluated in Section 5. Section 6 reports discussion and concluding remarks.

2 OVERVIEW OF RELATED WORK

This section presents a review of related work and offers the comparative analysis of existing blockchain related protocols related to cloud federation with the justification for the proposed architecture. Ferdous et al. [7] presented the blockchain based *decentralized runtime monitoring* architecture based on smart contracts to promote the accountability and transparency of access control decisions in cloud federations. Suzic et al. [17] introduced a security governance architecture, which permits the multi-layered, context and process-aware policy enforcement to meet the data security and privacy requirements in a heterogeneous environment of cloud federations.

Rahman et al. [13] argued that efficient scheduling is a significant issue for executing a performance-driven application such as workflows in cloud computing environment, but **the** existing techniques could not generate the schedules considering user QoS constraints and workflow level optimization. Consequently, they proposed an *Adaptive Hybrid Heuristic* for data analytics with the focus on satisfying the user constraints such as budget, deadline and data placement, while optimizing and minimizing the cost of execution.

Lee and Lee [10] noted that embedded devices would be used frequently in the Internet of Things context, but due to its limited capacities and resources, the substantial security properties have not been applied to the embedded devices yet. Therefore, they proposed a *secure firmware update scheme* that relies on blockchain technology.

Gaetani et al. [8] focused on the data integrity issues in the cloud federations. The authors proposed the two-layer blockchain based model for the cloud federations. The first layer is the permissioned blockchain, whereas the second layer is proof of work based blockchain. The proposed architecture provides the desired guarantees on data integrity, performance and stability, but doesn't focus on security issues.

Mizrahi et al. [11] proposed a protocol named *Proof of Activity*, a decentralized crypto currency network that combines the *Proof of Work and Proof of Stake*, to offer good security against possible attacks on Bitcoin. *Proof of Work* based protocol provides the decision-making power to the entities performing computational tasks, whereas *Proof of Stake* gives decision making power to entities holding stake in the system. Neither of these protocols are trouble free and can mitigate all the major threats the cryptocurrency faces, when applied on their own. Therefore, combining *Proof of work* and *Proof of stake* protocols can enhance the security against possible cryptocurrency attacks.

Christidis and Devetsikiotis [5] notes that a *Blockchain IoT* combination automates several existing, time consuming workflows in cryptographic verifiable manner. It also facilitates sharing resources and services leading to creation of marketplace for devices so that the devices can deploy various services at low cost. The study recommends using a new key for every

transaction or having a separate key for counterparty transactions to make pattern discovery difficult. The study also states that choosing the blockchain nodes wisely and having them sign the contract can minimize the collusion.

The growing trend of cloud federation imposes several challenges. None of the studies in the existing literature have adequately addressed all the challenges (stated in section 1) in a single study. Based on existing literature, we propose an updated cloud federation framework that caters to cloud federation needs related to security, integrity, and data quality in a single study.

3 PROPOSED FRAMEWORK

In this section, we introduce a cloud federation framework with the focus on secure communication channel processing the access requests and the access decisions between different components in cloud federation. The proposed architecture extends and builds on the blockchain-based cloud federation architecture proposed by Gaetani et al [8]. The proposed architecture is a two-layer blockchain-based architecture. The first layer is based on *mining rotation consensus algorithm. The mining rotation consensus* algorithm "divides the time into rounds, and for each round elects a member as a leader. The leader is in charge of receiving new operations, signing them with its private key and broadcasting them to other nodes in the network." Once all the minders sign the operations, they become part of the blockchain. At the *first* layer blockchain, every operation carried out in the distributed database is stored quickly and reliably. However, the *first* layer blockchain provides improved performance, but weak integrity guarantees due to lack of PoW. The *second layer* is designed as *PoW* based blockchain that stores the evidences of the database operations taking place at the first layer blockchain. The first and second layer interacts through blockchain anchoring technique. *Blockchain anchoring technique* is time- based operation that permits linking part of first layer blockchain with the block of second layer blockchain. At specific time intervals, a *witness transaction* containing the hash of first layer is sent to second layer block chain and these hashes are stored as the immutable and irreversible transactions. The second layer provides data integrity guarantees, but poor performance. But the principled interaction between first and second layer ensures data integrity guarantees and high performance.

However, other issues such as secure communications between access control systems and the federation members, privacy and QoS has not been addressed in the study. Therefore, in this study, we add new layer named infrastructure tenant, which will address security, privacy and QoS related issues along with data integrity guarantees and provide the cloud federations with a comprehensive solution to all the issues stated in section 1. Figure 1 depicts various components of the proposed cloud federation framework.

The framework comprises of three layers. The first layer is Infrastructure Tenant. The second and third layer have been borrowed from Gaetani et al [8]. The first and second layer is

based on *mining rotation consensus mechanism,* a permissioned blockchain. The third layer is PoW based blockchain and operates only in the background. All three layers interacts through blockchain anchoring technique similar to Gaetani et al [8].

3.1 LAYER 1: INFRASTRUCTURE TENANT

This layer is owned by all the members of the federation and is maintained virtually. The cloud federation has a single *infrastructure tenant*, which enables central functionalities underlying the federation. *Infrastructure tenant* is the virtual space formed by resources belonging to different clouds [14]**.** This layer is comprised of components described below:

3.1.1 ACCESS CONTROL SYSTEM

The access control system allows the cloud members to access services in a secure and controlled manner. It is based on the eXtensible Access Control Markup language (XACML) and is comprised of the *Policy Decision Point (PDP)*, the *Policy Enforcement Point (PEP),* the *Policy Retrieval Point (PRP),* and the *Policy Administration Point (PAP*), and *Policy Information Point (PIP). PDP* evaluates access decision based on the available policies. Upon receiving the user's request, *PEP* forwards the request to *PDP*, which calculates the access decision. The evaluation process carried by *PDP* depends, on the one hand, on access control policies made available by PRP *and* administered by PAP. On the other hand, it is based on contextual information provided by PIP [1].

3.1.2 WORK FLOW ENGINE

The Work Flow Engine creates, deploys and monitors the execution of the workflows in a cloud federation. Work Flow Engine has three sub-components. The Workflow engine component has been adapted from the architecture proposed by Rahman et al [13]. However, the activities of the subcomponents of the workflow engine have been modified in this study to fit the cloud federation operations. ***The Monitor*** monitors whether all the services are adequately executed or not. The monitor also depends on the event engine to gather the access logs and analyzes the gathered access logs to see if for the given request or the access is the expected one. ***Scheduler*** maps the tasks to the services based on the sophisticated scheduling algorithm while satisfying the user's constraints on cost and execution time. It prepares the service file based on the constraints imposed by the user and the availability of the service. The ***Dispatcher*** deploys the task to the corresponding service via the *Service Adaptor*. It also communicates with the miner requesting the service via Event engine.

3.1.3 SERVICE ADAPTOR

Service adaptor integrates federation services to the end user point and serves as the primary point of contact (by the dispatcher) for requesting the access code for members.

3.1.4 SERVICE RATINGS

Different cloud members may offer similar services, but at a different cost. All the services and the cost associated with them are reconciled together to find the optimal one, i.e., the best quality service at low price. The design of the rating service is out of the scope of this study.

3.1.5 EVENT ENGINE

Event Engine is responsible for intercepting and forwarding the data and the messages to create access logs. It also stores and keeps track of all the processes and events which occur due to the generation of all the messages from the user end and the infrastructure components.

3.2 LAYER 2: MINING ROTATION BASED BLOCKCHAIN

The mining rotation consensus algorithm "divides the time into rounds, and for each round elects a member as a leader. The leader is in charge of receiving new operations, signing them with its private key and broadcasting them to other nodes in the network." Once all the minders sign the operations, they become part of the blockchain.

3.2.1 LEDGER REPLICAS

Ledger replicas are the replicas of the distributed ledger held and updated independently by each member of the cloud federation. Miner in this framework refers to cloud federation member.

3.3 LAYER 3: PROOF OF WORK (PoW) BASED BLOCKCHAIN

It is the blockchain based on "Proof of Work." Proof of Work (PoW) is an obtuse mathematical puzzle used to validate the transactions and build new blocks to the chain. It is used by miners to compete against each other and get rewarded.

3.3.1 BLOCKCHAIN ANCHORING TECHNIQUE

It is time-based operation that permits linking the part of the first layer blockchain with the block of the second layer blockchain.

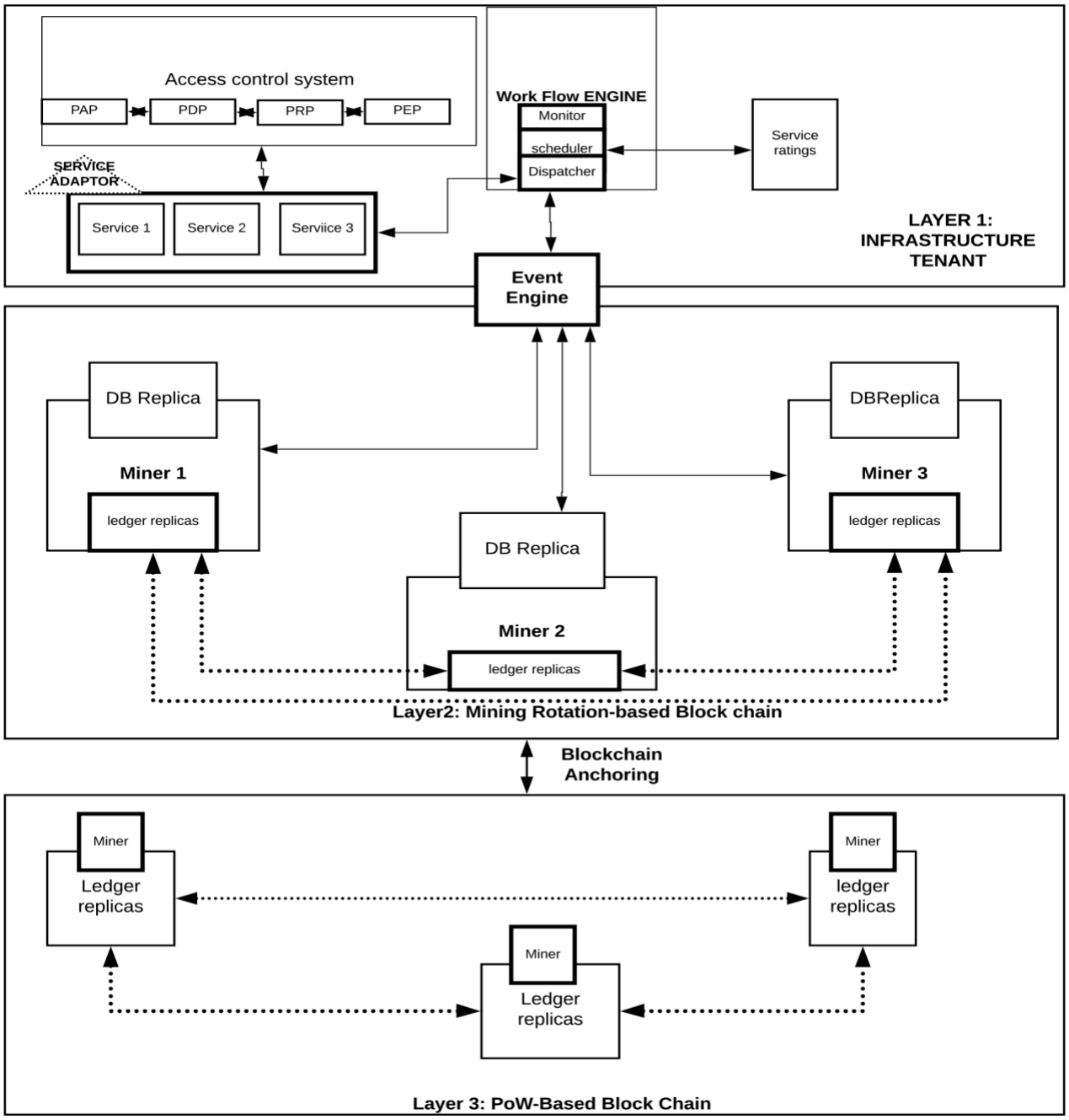

Figure 1. *Proposed Framework (Source: Layer 2 and Layer3: [8], p.152)*

4 DESCRIPTION OF THE PROCEDURE

The procedure (step 1-step 12) below provides the step by step description of communication between access control systems and the members of the cloud federation for processing access requests and access decisions. Notations used in the study are presented in Table 1.

Table 1. *Notations used in the study*

NOTATION	DEFINITION
req_ver_chk	*Request Verification Check Message*: This is the message sent by the request node (member) to the dispatcher (response node) to verify its identity and receive the service file. The request verification message includes the public key of the miner(member), identifier, Service Identifier, task identifier, service constraints, random number, and timestamp.
res_ver_chk	*Response Verification Check Message*: This is the message sent by the response node (*Dispatcher*) to the request node *(miner*) after verifying the identity of the Member (i.e., miner) The message includes the public key of the miner, service file, and the metadata of the service file.
SID_m	*Service Identifier:* This is the unique identification number of the service which a miner wants to access. For instance: resource services, analytics services, etc.
S_C	*Service Constraint*: This is the constraint imposed by the user on the execution of the service. It includes S_b(Service budget), S_d(Service Deadline) and DP_c(*Data Placement Constraint*).
ID_m	*Identifier*: This is the identification number provided to each member. The member can't operate without the service identification number.
S_r	*Service File*: The file is prepared by the scheduler based on the constraints imposed by the user and the availability of the service by the service provider. The file contains the documents related to the data placement as well as the service provider.
T_{ID}	*Task Identifier*: This is the identifier for a specific task such as computing, storage, processing in particular service.
M_r	*Metadata File* of S_r: The metadata file is composed of the file name, the hash value of the file, tracker URL, and length of the file [8]**.**
PU_m	*Public Key of the Miner* (member): The public key generated at the time of the request verification message. Every time a miner sends a request verification message, it is required to create a new public key. This is done to ensure the maintain the privacy on the blockchain.
PR_m	*Private Key of Miner* (member). It is the corresponding private key to the public key generated at the time of the request verification message.
PU_d	*Public Key of the Dispatcher*: Each time a dispatcher receives the request verification check message, it automatically creates the new pair of cryptographic keys: the public key and the corresponding private key.

PR_d	*Private Key of the Dispatcher*: It is the corresponding private key to the public key generated at the time of the request verification message.
PU_s	*Public Key of the Service Adaptor* that is available to all nodes in the blockchain network.
ID_d	*Identifier of the dispatcher*: It is the identification number provided to dispatcher.
ID_s	*Identifier of the Scheduler*: It is the unique identification number provided to the scheduler
r	Random number
$H(fS_r)$	*Verifier*: The hash value of the service file is generated according to the metadata of the service file. This value is used to ensure that the file has not been altered or corrupted [8]**.**
$Sign_m$	*Digital Signature of Member* (miner): The digital signature of miner is an encryption that uses the private key of the miner. Pubic key of the miner is added to the signature. Doing so will let anyone decrypt and verify the signature using the miner's public key [2].
$Sign_d$	*Digital Signature of the Dispatcher*.
req_access	*Request Service Access Message*: This message is sent by the miner (member) to request the service access code for a specific task within the specific service. It includes the verifier and the random number and is encrypted with the public key of the *Dispatcher*.
req_{SR_C}	*Request Service Access Code Message*: This message is sent by the *Dispatcher* to request the latest *Service Access Code* from the *Service Adapter*.
res_{SR_C}	*Response Service Access Code Message*: The *Service Adaptor* sends this message to the *Dispatcher*. It is encrypted with the *Dispatcher*'s public key and includes the latest *Service Access Code*, random number.
Sr_{ac}	*Latest Service Access Code*: The latest *Service Access Code* generated by the *Service Adaptor* based on the *Verifier* and the decision of the PDP. The *Service Access Code* is only valid for a particular period.
res_access	*Response Service Access Message*: The encrypted message generated by the *Dispatche*r to send the *Service Access Code* to the miner.

Step 1: Miner (request node) sends req_ver_chk i.e. *Request Verification Check* to the *Dispatcher* including its $PU_m, ID_m, SID_m, S_C,$ and r. Note: to prevent the replay attack, a random number r is generated. All the operations are logged in through the *Event Engine* which creates an initial access log.

$$req\_ver\_chk(PU_m \parallel ID_m \parallel SID_m \parallel T_{ID} \parallel S_C \parallel r) \quad (1)$$

Step 2: When the *Dispatcher* receives the res_ver_chk *message*, it first communicates with the *Scheduler* through its unique identification number i.e. ID_d and obtains the S_r (i. e. , the $Service\ File$) and the corresponding M_r, (i.e., *metadata* of the Service File) from the *Scheduler* based on the $PU_m, ID_m, SID_m, T_{ID}, and\ S_C$.

Note: The communications within the infrastructural tenant cannot be compromised, because the intra -tenant communications rely on secured VPN tunnels [7].

Step 3: Now, the *Dispatcher* responds to the miner by sending encrypted (encrypted with the public key of the miner) *Response Verification Check* (res_ver_chk) message including S_r, the corresponding M_r, *r+1,* his/her public key i.e., PU_D and his/her identifier i.e. ID_d.

$$res\_ver\_chk(E(PU_m, S_r, M_r \parallel r+1 \parallel ID_d \parallel PU_D)) \quad (2)$$

Step 4: When miner receives the res_ver_chk message, it decrypts the message with its private key (i.e., PR_m) and checks r+1. The miner then generates the *Verifier* $H(fS_r)$ from M_r.

***Step 5*:** Then the *miner* encrypts the ID_m, $H(fS_r)$ *and r+2* with the PU_d and generates the digital signature $Sign_m$ using his/her private key (i.e. PR_m) over a $req\_\_access$ message containing the encrypted data i.e. ID_m, $H(fS_r)$ *and r+2*

$$req\_\_access(E(PU_d, ID_m, H(fS_r) \parallel r+2) \parallel Sign_m) \quad (3)$$

***Step6*:** When the *Dispatcher* receives the $req\_\_access$ message, it first verifies the message's integrity, and originator with the digital signature $Sign_m$ attached to it by decrypting the message with the public key PU_m obtained from req_ver_chk (recall Step 1). Since ID_m, $H(fS_r)$ *and* $r+2$ has also been encrypted with the private key of the dispatcher, the dispatcher decrypts the encrypted data with his/her PR_d and checks $r+2$

Step 7: Now, the Dispatcher communicates with the service adaptor through its unique identification number and sends the *Request Service Code* message (i.e., req_{SR_C}) to the *Service Adaptor* by encrypting the $H(fS_r)$, *r+3* with PU_S (i.e., the public key of the *service adaptor)*.

$$req_{SR_C}(E(PU_S, H(fS_r) \parallel r+3)) \quad (4)$$

Note: *The miners* can't access the *service code* directly from the *Service adaptor* since the *Dispatcher* has to verify the identity of the miner and the *Verifier* generated for the *service file* before sending the request for the service access code to the *Service adaptor*.

***Step 8*:** The *Service Adaptor* decrypts the message with its private key and communicates with the access control system through its unique identification number i.e. ID_s **Note**: The legitimacy of the access control system is assumed because no updates(operations) are complete until all the members sign the operation. The service adaptor prepares the Sr_{ac} *(the service access code)* based on the PDP decision and the verifier $H(fS_r)$.

Step 9*:* The *Service Adaptor* communicates with the dispatcher through ID_s and sends the *Response Service Code Access* message (i.e., res_{SR_C}) including the Sr_{ac} (i.e., the current *service access code*) and *r*+4 that has been encrypted with the public key PU_d of the dispatcher.

$$res_{SR_C}\big(E(\,PU_d\,, Sr_{ac} \parallel r+4\,)\big) \tag{5}$$

Step 10: The *Dispatcher* upon receiving the message checks the r+4 and decrypts the *Response Service Code Access* message res_{SR_C} with PR_d.

Step 11**:** Later, the *Dispatcher* sends the *Response Service access message (i.e.*, res_access) to the miner. The message includes Sr_{ac}, and r+5 which has been encrypted with PU_m(i.e., the public key of the *miner*). The dispatcher attaches the digital *signature* ($i.e., Sign_d$) to the *Response Service access message*. The *dispatche*r uses his/her Private key (i.e.*,* PR_d) to generate the digital signature.

$$res\_access\ (E(\,PU_m,\ Sr_{ac} \parallel r+5) \parallel Sign_d)) \tag{6}$$

Step 12**:** The *miner* upon receiving the *Response Service Success message* verifies the message's integrity and originator by decrypting the message with the public key of dispatcher previously shared (recall step 3). Later, the *miner* decrypts the Sr_{ac} with his/her public key P_m and checks (r+5). Now, the miner can access the service from the service adapter by using the Sr_{ac}

Note: Each time a miner requests a new service access, he/she has to generate new public/private keys and so as the dispatcher and scheduler for responding to new service request.

5 A CLOUD FEDERATION ILLUSTRATION USING THE PROPOSED FRAMEWORK

Peffers et al. [12] reports that several studies that involve the evaluation of framework have used illustrative scenario. Similar to those studies we are also going to use the illustrative scenario for the evaluation of our artifact. This scenario is based on the SUNFISH FaaS case study, in which the concept of Federation-as- a-service (FaaS) was introduced. The *Sunfish FaaS* is an innovative cloud federation platform that permits the public and private clouds to federate themselves to share the services, resources, and data [13]. In this paper, *FaaS functionalities* take place at the first layer, *service request* and service *provider* functionalities take place at the second and data *governance* takes place at all the three layers. At the second layer, miner 1, miner 2 and miner 3 represent each cloud member respectively. For the sake of convenience, we will only illustrate the service request functionality and data governance of FaaS.

We assume that service 1, 2 and 3 represents *resource services, analytics services,* and *software deployment services* in the proposed framework. Each *service* has various tasks associated with it. Miner *1* wants to utilize the storage service for its clients as it doesn't possess the storage resources. *Miner 2* and *Miner 3* are *service providers* of the storage services. Once Miner 2 and 3 register their services in cloud federation, the *service publication* publishes the

services and makes the *service list* available for *Miner* 1, 2 and 3. Each *service* has a *unique service identifi*er. *Miner 2* and *Miner 3* both provide the storage resources, but at different cost and speed. The *service ratings* functionality rates the services provided by *Miner 2* and *Miner 3* based on the service speed and corresponding cost for executing the task in a service. We also assume that the *service list* and the *corresponding tasks list* including the *Service identifier* and *Task identifier* is made available to each member while registering for the cloud federation.

Step 1: *Miner 1* broadcasts the *request verification chec*k message along with its *identifie*r, *Service identifier, task identifier, Service constraints*, his/her *public key*, and the *random number* to the blockchain network. When the *request verification check* message is sent, time *stamp* is generated automatically. The *service constraints* imposed by *Miner 1* includes data *placement constraint* and budgetary *constraint* for executing the whole service.

Step 2: When Dispatcher receives the *request verification check* message, it verifies identity of miner based on *identifier* of miner stored in its database. If *miner's identifier* equals identifier of miner stored in the *database* of *Dispatche*r, *Dispatche*r will communicate with the *Scheduler* to obtain the Service file and corresponding Metadata file, otherwise, the process is terminated with an error.

Note: *Scheduler* prepares the *Service file* based on the *service constraints* imposed by the *Miner1*. The *Dispatche*r obtains the *Service file* and the *Metadata file* from the *Scheduler*.

Step 3: *Dispatcher* sends the *Response verification check* message including the encrypted (encrypted with the public *key* of *Miner 1)* new Service file generated by *Scheduler, Metadata of the Service file,* r+1, his/her identifier i.e., identifier of the dispatcher, *his/her* newly generated public key i.e., dispatcher's public key. The *Response verification check message* can only be decrypted with the corresponding *private key* to the *public key* of Miner1.

Step 4: *Miner 1* upon receiving *Response Verification Check* decrypts the *Response Verification Check* message with his/her *private key i.e.* private key of miner 1 and checks the r+1. The *Miner* 1 prepares the hash value of the *Service* file, i.e., the *verifier* from the *Metadata file* of the *Service file* [10].

Step5**:** *Miner 1* encrypts the *verifier* of service file and r+2 with the *Public key* of *Dispatcher*. Now Miner 1 generates the digital signature using his/her (Miner) private key (i.e., The corresponding private key to the public key shared in step1) and attaches *digital signature* over the *Request Service Access* message containing encrypted identifier of the miner, verifier of the service file and r+2.

Step6**:** When Dispatcher receives the *Request service access* message, it decrypts digitally signed message *using* the *public key of the Miner 1* obtained from the *Request verification message* in step1 to verify integrity and originator of the message. The successful decryption accounts a digital signature verification meaning that there is no doubt that the message was sent by the miner 1. Now the *Dispatcher* decrypts the identifier, verifier of service file and r+2 with his/her

private key i.e., private key of the dispatcher. The verifier helps dispatcher to verify the integrity of the *Service file*.

Step 7: Now, the *Dispatcher* communicates with *service adaptor* using its (i.e., dispatcher's) unique identification number and sends the *Request access code* message by encrypting the verifier of service file and r+3 with the public key of the *Service adaptor* to request the *Service code* for *Miner 1*.

***Step 8*:** The *Service Adaptor* upon receiving the *Request access code* message decrypts the message with his/her *Private key* i.e., Private key of the service adaptor. The *Service Adaptor* communicates with the *PDP* via its unique identification number and prepares the *Service access code* based on *PDP* decision and *Verifier* of the service file. The Service access code will be the current *Service Access Code.*

Step9: Now, *Service Adaptor* encrypts the *Service access code* with the *Public key* of *Dispatcher* and sends the *Response service access code* message to dispatcher including the *Service access code* and r+4.

Step 10: *Dispatche*r receives the *Response service access code* message. The *Dispatcher* decrypts the *Response service access code* message using his/her *Private key* and checks r+4.

***Step 11*:** *Dispatche*r encrypts the *Service access code* and r+5 with the *Public key* of Miner *1*. *Dispatcher* also prepares the *Digital signature* using his/her (dispatcher) *Private key. Later, the digital signature* of *Dispatcher* is attached to the *Response service access* message. The *Dispatche*r sends the Response *service access* message including encrypted *service access code and r+5 to Miner 1.*

Step 12*: Miner* 1 decrypts the message with previously shared *Public key* of *Dispatcher* (recall step3) to verify the integrity and originator of the message. Later, *Miner 1* decrypts the *Response service access* message using his/her *Private key.* Now, Miner 1 can access the storage resources of the miner 2 based on the *Service access code*.

6 DISCUSSION AND CONCLUSION

This section discusses how the solution we proposed successfully addresses issues outlined in section 1. The *Scheduler* prepares the service file based on QoS constraints imposed by miner. The *Scheduler* interacts with the service ratings to get the information on service quality and cost associated with service provided by each member. The *Scheduler* maps the service constraints to the information provided by the service ratings to choose the best available service for the members and hence addresses (**Issue 1)**.
In Section 4, Step 1 - Step12 exhibits secure communication between members and access control systems. Inclusion of various components such as identifier of the nodes, public key, verifier, digital signature, service access code, encryption, and random number provides enhanced privacy and authentication and hence address the security issue **(Issue 2).** The

components are required to use new cryptographic key each time a new service access is made, making the pattern identification difficult for the interested parties/attackers and enhance the privacy. (**Issue 7**)

Also, all the events (from step 1 and step 12) are recorded in *Event Engine*. The *Monito*r collects logs from the event engine to see if, for the given request, access is the expected one. Since, all miners use their private keys to sign the operations, the attacker will need to steal all the private keys of the participating members to alter databases **(Issue 3*)***. In the setting of cloud federation, it would mean attacking multiple distributed cloud members simultaneously.

The *consensus algorithm* attributed at the first and second layer considers all the members in the federation. Therefore, no database operation can be completed without all the members being aware of it. **(Issue 4)** [8]

If multiple members collude together to attack a single *miner*, then the honest *miner* can react by not sending its message within consensus protocol and prevent the malicious database operation from being completed. **(Issue 5*).*** If collation attack was targeted to modify the information already stored in first and second layer i.e., the information is already agreed on by all the federation members, then the honest *mine*r can show the original version by presenting the messages previously signed and sent by other members (i.e., an information is only stored, when the consensus on the message is received) [8].

To resolve the performance issues **(Issue 6*)*** first and second layer is based on *mining rotation algorithm* and leverages *PoW* only in the background. The operation on the database is completed as soon as it is elaborated by first and second layer blockchain. At *first*- and second-layer block chain, every operation carried out in the distributed database is stored quickly and reliably. However, first and *second* layer block chain provides improved performance, but weak integrity guarantees due to lack of PoW. The *third layer* is designed as PoW based block chain that stores the evidences of database operations in first and second-layer block chain. These evidences are stored as the immutable and irreversible transactions, but with the poor performance [8]. Overall, the principled interaction between three layers permits achieving effective assurances on data integrity, security, privacy and enhanced performance. This study identifies the requirements of the cloud federation by studying the prior literature. None of the studies (as per our knowledge) have explicitly integrated all the challenges related to cloud federation within a unifying framework. Our first contribution is combining all the issues associated with cloud computing specifically with reference to cloud federation in one study. The second contribution is an innovative blockchain based framework that permits secure communication between the members of the federation and the access control systems and ensures privacy. The third contribution is that the proposed solution solves the QoS issue by considering service constraints imposed by members, while providing the services. We also evaluated the proposed framework through the illustrative scenario. One limitation to this study is that availability issue can emerge, if a single miner refuses to sign the operations. The study can be extended in different directions. First, the further research may be conducted by deploying the proposed framework in some simulated setting. Second, future research may focus on

integrating availability with data integrity as well as secure access controls to overcome the limitations of our framework.

REFERENCES

1. Anderson, A., Nadalin, A., Parducci, B., Engovatov, D., Lockhart, H., Kudo, M., ... & Moses, T. (2003). extensible access control markup language (XACML) version 1.0. *OASIS*.
2. Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.".
3. Bertino, E. (2016, June). Data security and privacy: concepts, approaches, and research directions. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual* (Vol. 1, pp. 400-407). IEEE.
4. Chen, D., & Zhao, H. (2012, March). Data security and privacy protection issues in cloud computing. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on* (Vol. 1, pp. 647-651). IEEE.
5. Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *Ieee Access*, *4*, 2292-2303.
6. Encryption, P. K. (2004). Digital Signature: How do they work. *CGI group Inc-2004*.
7. Ferdous, S., Margheri, A., Paci, F., & Sassone, V. (2017). Decentralised runtime monitoring for access control systems in cloud federations.
8. Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2017). Blockchain-Based Database to Ensure Data Integrity in Cloud Computing Environments. In *ITASEC*(pp. 146-155).
9. Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., & Kunze, M. (2011). Cloud federation. *Cloud Computing*, *2011*, 32-38.
10. Lee, B., & Lee, J. H. (2017). Blockchain-based secure firmware update for embedded devices in an Internet of Things environment. *The Journal of Supercomputing*, *73*(3), 1152-1167.
11. Mizrahi, I. B. C. L. A., & Rosenfeld, M. (2014). Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake.
12. Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012, May). Design science research evaluation. In *International Conference on Design Science Research in Information Systems* (pp. 398-410). Springer, Berlin, Heidelberg.
13. Rahman, M., Li, X., & Palit, H. (2011, May). A hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment. In *Parallel and Distributed Processing Workshops and Ph.D. Forum (IPDPSW), 2011 IEEE International Symposium on* (pp. 966-974). IEEE.
14. Schiavo, F. P., Sassone, V., Nicoletti, L., & Margheri, A. (2016). Faas: Federation-as-a-service. *arXiv preprint arXiv:1612.03937*.
15. So, K. (2011). Cloud computing security issues and challenges. *International Journal of Computer Networks*, *3*(5), 247-55.
16. Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, *34*(1), 1-11.
17. Suzic, B., Prünster, B., Ziegler, D., Marsalek, A., & Reiter, A. (2016, October). Balancing utility and security: Securing cloud federations of public entities. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 943-961). Springer International Publishing.