Wireless Networks (2020) 26:4943–4954
https://doi.org/10.1007/s11276-019-01968-x

An extended access control model for permissioned blockchain frameworks

Muhammad Yasar Khan[1] · Megat F. Zuhairi[1] · Toqeer Ali[2] · Turki Alghamdi[2] · Jose Antonio Marmolejo-Saucedo[3]

Published online: 4 March 2019

Abstract
In distributed environment, a digital transaction or operation requires transparency and trust among multiple stakeholders. Several approches address such issues however, among these blockchain provides a viable solution which has received wide acceptance in the recent past. Permissioned blockchain solutions adopt more efficient consensus algorithms and smart contracts. There are many smart-contract solutions exists (such as, etherium, IBM blockchain, hyperledger fabric), however, much of them mainly follow traditional access control models. A role-based access control model provides controlled access of resources to members. This research work presents an extended usage control model known as DistU (Distributed Usage Control). DistU is proposed to capture all possible access control models required by a business for permissioned blockchain frameworks. DistU can monitor a resource continuously during the operation and update the attributes accordingly, performing different actions, such as denying or revoking permissions. We believe that the proposed DistU usage control model can provide a fine-grained control for blockchain resource management. The paper also contributes to provide a protoype implementation of fine-grained permission model on Hyperledger Fabric. The reason of selecting Fabric for this research is that, it is the first execute-order achitecture blockchain that provides a platform to develop general business applciations. Secondly, it is an opensource operating system of permissioned blockchain with huge industry support.

Keywords Permissioned blockchain · System chaincode · Access control · Usage control · Continuous monitoring

1 Introduction

The conception of blockchain is introduced by an anonymous researcher in 2009 [1]. The initial idea of such technology is to offer a digital currency that is completely decentralized. By using cryptography techniques and consensus algorithms [2], known as Proof-of-Work (PoW) [3], a trust paradigm between untrusted members can be developed. Each transaction of the block must link to its previous generated block, and because of this structure, a blockchain can traverse back to the origin block. As such, a blockchain contains immutable records of every transaction made.

✉ Megat F. Zuhairi
megatfarez@unikl.edu.my

1 Universiti Kuala Lumpur, Kuala Lumpur, Malaysia

2 Islamic University of Madinah, Medina, Saudi Arabia

3 Panamerican University, Mexico City, Mexico

Beside the cryptocurrency, Blockchain functionality can be adapted to any distributed business environment. Bitcoin established the building blocks for a new era of computing. However, Bitcoin is a public blockchain and works on a PoW consensus algorithm. The PoW is a highly scalable algorithm, but the provision is suboptimal because of inefficient transaction execution. According to the current state of Bitcoin, the system executes 7–8 transactions in a second, which is not acceptable in general purpose business transactions [4]. Due to such inefficiency, many IT and financial enterprises as well as other organizations are looking into a new blockchain model that is suitable for general-purpose business applications.

In 2015, IBM started Hyperledger, which is an umbrella project hosted by The Linux Foundation [5]. It serves as a hub for open industrial blockchain development. Currently, there are five projects running under Hyperledger, including Hyperledger Fabric, Swatooh, Iroha, Burrow, and Indy. In this research work, the author adopts the Hyperledger Fabric, which is a modular and expendable open source system for utilizing permissioned blockchain. It is currently used in many proofs-of-concepts of distributed ledger technology, production systems, and prototypes in different industries. Blockchain offers a new layer known by smart contract or chaincode to deploy an organizational business transaction policy. Smart contracts allow a programmable method facilitating, verifying, or enforcing the negotiation of business transactions. Additionally, Hyperledger Fabric is a modular, decentralized operating system and it support multiple pluggible scalable consensus algorithms.

Hyperledger Fabric, uses Role Based Access Control (RBAC) [5, 6] as its permission model. This model governs user access to a system's resources (assets), based on roles. RBAC does not focus on the ongoing access and attribute aspect of permissions. In such model, continual access and attribute updates are required in many business scenarios. For example, RBAC enables a user U of role A to access a specific object. However, the approach does not have a mechanism to monitor an object continuously or to revoke access. In a distributed environment, a finer control over the usage of digital objects as compared to the conventional access control such as RBAC, is required.

2 Related work

This section presents previous research studies related to the Blockchain security and privacy.

Ronghua Xu et al. [7] conducted a survey on the security and privacy of blockchain in the field of Internet-of-Things (IoT). It is deemed that the existing centralized authentication system is not feasible for blockchain architecture. The research work proposed a security model called BledCAC, which recommend capability delegation process to propagate permission in IoT, using blockchain infrastructure.

To address the aforementioned problems, a Federated Capability-based Access Control model (FedCAC) [8] is proposed. This research identifies the important aspects of the Access policy but fails to address any solution with regards to resources dynamic usage and prevention of its excessive use.

Jason Paul Cruz et al. [9] proposed a model called Role-Based Access Control using Smart Contracts (RBAC-SC). In this model, the author makes use of Ethereum's smart contract technology to realize a trans-organizational utilization of an organization's roles. In this model, the trans-organizational behavior of the user's roles is presented. For instance, a student of university A is allowed to buy books from a shop S. Both are different organizations, but the shops accept each other's role up to some certain extent. The research work provides the cross-organization role usability concept, which is very unique but lacks the perspective of role invocations, which is the inherent problem in any RBAC based model. In short, it is not possible to restrict usage access of a role unless it is manually revoked.

Aissam Outchakoucht et al. [10]. explain the importance of the user data on the blockchain network. It emphases strongly on the importance of data privacy as it may contain very confidential data. The author proposed Machine Learning (ML) techniques to the blockchain network which will ensure optimized self- adjusted security policy. The proposed model is profound but handling access decision to a central authority eliminates the end-to-end security mechanism, which is the core of blockchain concept. The second problem in the proposed scheme is it does not impose rule restriction to the user and allows ML algorithms to decide access to the resources. In short, it does not allow resource access customization according to the need of an organization but rather using an automatic algorithm.

3 Background

Most blockchain projects emphasize transactions and consensus algorithms. Minimal work is done on user access policies in terms of obligations and usage alerts. Currently, most blockchains entertain authorization policies only when an explicit request is made regarding an resources. Smart contracts address organizational business policies; however, the developer would not fully cover every usage control aspects of the resources within the model. The inherent problem of blockchain access control is summarized below.

Existing access control models in permissioned blockchains only authorizes one-time access. However, after access is granted, no mechanism exists to subsequently control resource's access, for example, to continuously monitor and revoke permission.

In this research work, an improved model is proposed, which contributes to the usage (continual access) and access control of the resources as defined in the Hyperledger Fabric architecture. The work proposes a fine-grained security model to the blockchain infrastructure, which covers all traditional access controls, such as discretionary access control (DAC), mandatory access control [11] (MAC), and (RBAC). It also extends to digital rights management and other method of modern access control management required for businesses.

4 Blockchain

A blockchain can be defined as an immutable ledger for recording transactions, maintained within a distributed network of mutually untrusting nodes/peers. Every node in the network maintains a copy of the ledger. When a new transaction is generated, each node first verifies its validity using a consensus protocol, and then it is grouped with multiple transactions to a block and builds a hash chain over the blocks and broadcasts to the entire network. Such process forms the ledger by ordering the transactions, which is necessary for consistency. Blockchain emerged with Bitcoin and is generally viewed as a promising technology to run trusted trades in the digital world.

In a distributed system, various autonomous components work together to form a single system. Such distributed systems enhance the performance, reliability, scalability, and availability of resources in the network. These components can be of different processing capabilities and can be located at any geographical location. In a distributed system, information is shared or replicated to each node of the network. Any node can generate a new transaction that changes the state of data. A distributed system uses consensus algorithms to register any new transaction in a specific order to the network. Two popular consensus algorithms are used, namely Raft and Paxos. In both algorithms, the system selects a leader through leader election, which is responsible for committing a transaction and keeps track of its order. In distributed network algorithm, some limitations are that all members should be trusted, they must rely on the selected leader, network scalability is very hard and its architecture is very fixed to be customized.

Blockchain is a linked-list of data blocks, which contain a pointer to the previous block hash. Blockchain uses a hashing technique to maintain the integrity of the data blocks. Hashing is a mathematical function that transforms arbitrary input data to a fixed-size digest value. It is computationally complex to reverse a hash function value. In other words, if a hash function f for an input i produces hash value f (x), then it should be very difficult to find any other input value j, such that f (i) = f (j). Figure 1 shows blockchain ledger structure.

Blockchain are mainly of two types i.e. Public/Permissionless and Private/Permissioned blockchain.

- *Permissionless blockchain* In Public blockchain any anonymous user can join the network. This model is best suited for Business-to-Consumer (B2C) and Consumer-to-Consumer (C2C) model's use cases. Most of these systems use cryptocurrency to simplify the exchange of values among members of the network.
- *Permissioned blockchain* A permissioned blockchain is a closed ecosystem in that each member provides the credential to become the part of the network. This type of blockchain is built to allow an organization or a consortium of organizations to efficiently exchange information and record operations. Permissioned blockchain follows Business-to-Business (B2B) model. Hyperledger Fabric is type of permissioned blockchain.

Ensuring trust of processing operations in public and private blockchain is very different. As compared to public blockchain private blockchain needs more control our its shared resources or data. In this paper we highlight existing security structure of the Permissioned blockchain and also proposed a new security model that incorporate all the required security constraints for a business solution. We opt Hyperledger Fabric permission model for our proposed architecture.

5 IBM hyperledger fabric

Hyperledger Fabric is an open source permissioned blockchain framework. Similar to other blockchain technology, it works in a distributed peer-to-peer network topology. Fabric intends to provide a modular and scalable architecture that can be deployed in many industries e.g.

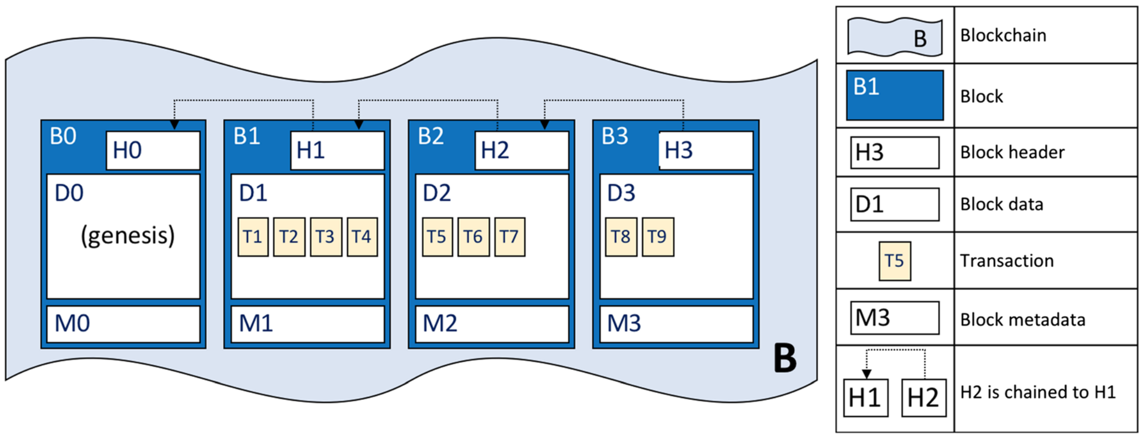

Fig. 1 Blockchain ledger structure [12]

from banking to healthcare and supply chains [3]. language. In contrast to other blockchain, Fabric introduced novel concept of execute-order architecture instead of order-execute model [4]. By using this model, transaction throughput is increased significantly. Bitcoin and Ethereum (public blockchain) runs 8–20 transactions per second (tps) while Fabric can run 3500 + tps.

In Fabric, multiple peers combine to form a private communication network called channel. Each channel maintains its own distributed-network, organizations, distributed-ledger, members, chaincode(s) and orderer service(s). A single node can join multiple channels, keeping each channel information isolated from other. Each peer in the network is considered as Node. Below is the list of Hyperledger Fabric main components. Fig. 2 shows overall fabric network structure:

- *Channel* Hyperledger Fabric works on channel, channel combine set member to a network. One channel data is confidential from another channel. If a scenario is need to secure communication of two or more member in the same channel, fabric allow us to create sub-channel or to create a private data section within a channel. Using MSP fabric can issue identity certificates for n-level of channel network.
- *Membership service provider (MSP)* In Hyperledger Fabric members are assigned unique id which is used in X.509 digital certificate. Fabric issued different set of certificates for different members which are linked to different roles in the network. Through MSP we can defines Intermediate CAs (in case of large network), Organizational Units, Revoked Certificates, Node Identity, KeyStore for Private Key,TLS Root CA and TLS Intermediate CA.
- *Chaincode* Chaincode is a piece of code that defines a business logic of an application. chaincode are install and initiated on peer level of a channel. Chaincode are used to update state of the resource in the DSL [14]. Hyperledger fabric uses Shim API layer to protect DSL from unauthorized access. Chaincode functionality extends Shim init() and invoke() interfaces to perform get or put operation on the ledger. Chaincode are mainly used in two scenarios i.e. to define business contracts or to manage digital resources on a network. Chaincode are further divided into two types system chaincode and user chaincode i.e.
  - *System chaincode* System chaincode are used to monitor user chaincode life cycle it also ensure validity of the transaction proposal. system chaincode can be customized and build for different business purposes. By default we have some system chaincodes enabled on a pper i.e. Endorsement System Chaincode (ESCC) handles endorsement of a transaction, Validation System Chaincode (VSCC) is responsible for checking the validity of the transaction and (LSCC) and Query System Chaincode (QSCC). system chaincode runs as part of the peer process and has the authority to ACCEPT or DENY a transaction proposal.
  - *User chaincode* User chaincode are defined by application developers. User chaincode runs in a separate docker container and linked with the channel network. User chaincode initiate transaction proposal proposal.
- *Consensus algorithms* Consensus defines rules of the game, it make sure of how an ecosystem will arrive at a single view of the ledger. Generally consensus are used to ensure the right order of the transaction occurring. Hyperledger fabric along with the correct order of the transaction also ensure validity of the transaction and also perform other custom endorsement policy before committing it to the DSL.
- *Endorsement policy* Hyperledger Fabric introduced a new concept of Endorsement Policy (EP) that is defined

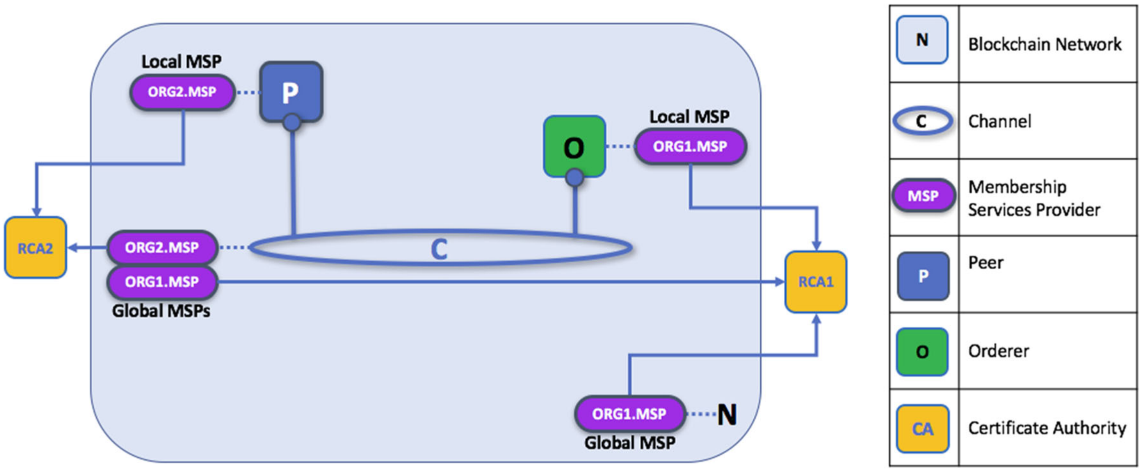

Fig. 2 Hyperledger fabric architecture [13]

over the execution of the chaincode. EP support the execute-order Fabric architecture and check validity of the proposed transaction before it is committed. For a transaction to be accepted in the ledger it must be executed and validated on the mentioned endorser node and upon their approval it would be added to the ledger. EP make use of the roles specified by the MSP in the network. e.g. Org1.admin defines administrator of the network, Org2.member defines any member, Org3.-client defines the client or organization and Org4.peer defines the peer of the organization 4, Endorsement policy are further divided into two types.

- *Chaincode level endorsement* Chaincode level endorsement are defined with a -P switch while installing a chanicode on a channel. Command use for EP specify list of the organization that will be needed to execute and validate propose transaction. EP allow us to define different combination of the organization using AND and OR operators e.g. *AND('Org2.peer', 'Org3.peer')* or *OutOf(1, 'Org2.peer', 'Org3.peer')* or *OR('Org2.peer', 'Org3.peer').*
- *Key level endorsement* Key level endorsement are applied at the function level within a chaincode. Upon arrival of the new transaction Fabric First check for the key level endorsement policy associated with a function and if there is no policy then it looks for the chaincode level policy. Key level endorsement allow us to select different set of the organization or members from the chaincode level endorser.

* *Ordering peer or orderer* Create Blockchains blocks (by defined size etc.), determine the order and broadcast to all nodes (in a channel) for an update in the local database. Generally, a Fabric contains two types of databases on each peer which is shown below.
* *World state* It is a version key-value pair database such as CouchDB/LevelDB which stores the current state of the resources. It also shows the state of the system at any point in time. The world-state is used to reduce traffic on the actual ledger.
* *Ledger* It is the immutable chain of the hashable blocks (World-state database can be restored from this ledger at any time, as it contains all the transaction of resources from start to final state).

6 UCON adoption for hyperledger fabric

The most popular model of usage control is termed as Usage Control (UCON), which was proposed by Park and Sandhu [15] and later formalized by Zhang et al. [16]. The model is based on two core concepts:

* Decision continuity
* Attribute mutability

Decision continuity refers to the concept described in previous section, i.e. decision to grant access is not a singular operation. It is a continuous action that is carried out in parallel to the subject's access. After access has been granted, the UCON models reference monitor keeps track of usage and can perform several tasks based on the usage.

Attribute mutability is the second aspect of UCON that enables changes to a subject or objects attributes as a result of the attempt to access an object or actually accessing an object. The UCON model allows policy writers to define certain types of attribute Mutability. Mutability of attributes and continuity of decisions is illustrated by Fig. 3.

There are various states in UCON i.e.'initial', 'requesting', 'denied', 'accessing', 'revoked' and 'end'. The model puts each s, o, r triplet variable in one of the states where s is the subject, o is the object and r is the right. Therefore, if

$$t1 = (s1, o1, r1) state(t1) = accessing$$

Based on the code, it can be deduced that the subject s1 is currently exercising right r1 on object o1 and the reference monitor is keeping track of this usage.

In addition, the reference monitor is operational when s1 attempts to access o1. This is the non-bypass ability feature of the reference monitor and such method ensures that each access is mediated by the security policy enforced by the reference monitor. Immediately after access to o1, the state of the triplet is changed to 'requesting'. The reference monitor can perform several things at this stage.

* *permitAccess* It can allow s1 to access o1. In this case, the state changes to accessing.
* *denyAccess* It can decide to prevent access thus resulting in the state changing to denied. One of these two decisions have to be made by the reference monitor which are mutually exclusive.
* *preUpdate* The reference monitor can also optionally update some attributes of the subject or the object. This is termed as a pre-update operation and is highly useful in keeping track of, for example, how many times a subject has tried to access a subject. Hence, repeated attempts by a subject to access a restricted object can be detected by the reference monitor and later the subject can be penalized.

If the subject is denied access, the session finishes and the triplet have to return to the initial state before performing any restricted action. On the other hand, if the state changes to accessing, the subject is free to access the object. However, at any point in time during access, the

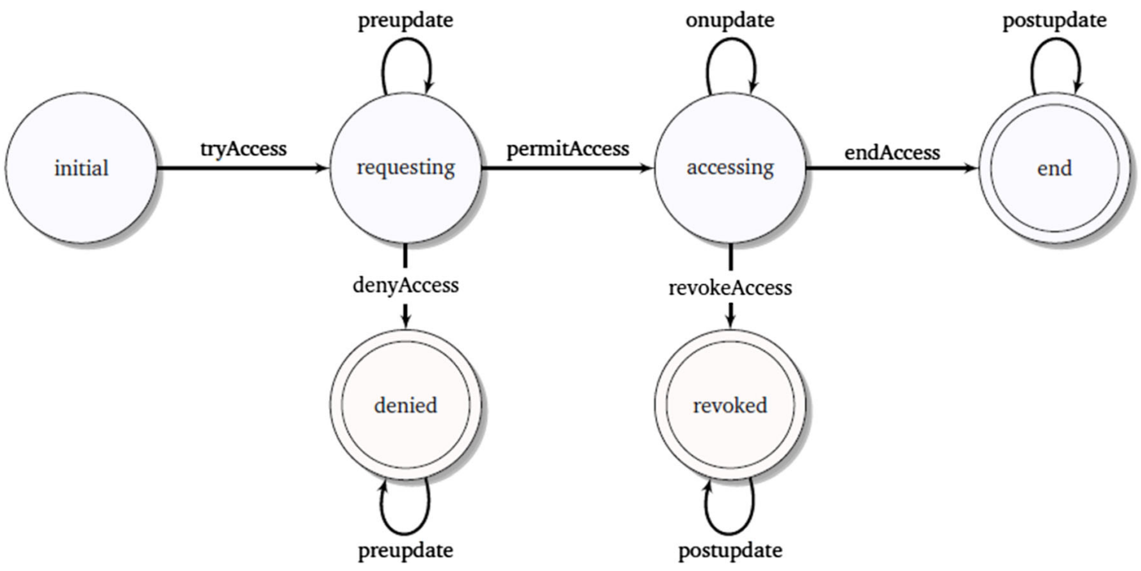

Fig. 3 UCON model states

reference monitor might revoke the granted permission. There are four operations that can potentially happen.

- *endAccess* The first one is that the subject voluntarily gives up the object i.e. finishes accessing it. In such case, the reference monitor does not have to perform any task in the accessing state.
- *revokeAccess* The second, it may be possible that as a result of the usage (or due to changing of the operating environment), the reference monitor decides to revoke the access decision previously made. Subsequently, the subject is no longer able to access the object and the state of the system changes to 'revoked'.
- *onUpdates* During access, the reference monitor may update some attributes to keep track of system usage. This is a clean method of gauging usage.
- *postUpdates* Optionally, the reference monitor can perform different updates to subject or object attributes based on whether the subject voluntarily gave up the object or if it was forcefully revoked.

In the following section the UCON model is defined for fabric normal access control as well as usage control.

7 DistU: extended permissions model for hyperledger fabric

Usage control enhances the access control systems by introducing the idea that an access control must not only be limited to granting access to an object and exercise the right. In addition, it should also consistently keep track of the usage of objects and determine whether to continue allowing the subject's access over objects. If a subject exceeds the allowed quota or if another constraint is violated during the use of the object, the allowed right must be revoked from the subject at runtime.

The idea is simple in nature and enables different policy specifications and thus extends the expressiveness of access control systems. With such features, it is possible for an organization to specify a policy that decides whether an object can be accessed only within the premises and within a certain time frame. It also allows specification of constraints such as the permitted usage time per session and revocation of usage after the specified limit has reached.

7.1 Model definition

Hyperledger Fabric architecture is based on the concept of Resources, Members, Operations, and Conditions. According to the proposed usage control for Fabric, models are defined based on the new requirements. The models are an extension to the core Hyperledger Fabric and can assist development of new future business applications based on blockchain.

a. *Definition 1.* (Members and Assets) This denote the set of member in an active business network with M and set of all resources within a transaction **A**. Resource in a specific operation can be accessed by the member association function **ζ:** R → M. Each resource is associated with a unique member.
b. *Definition 2.* (Operations) A set of all permission, is denoted with '**O**', while M is labeled for members. Both are specified as a collection of labels for accessing a resource. Operation requires function $\mathbf{M_s}$: R → R to provide the access $\mathbf{o} \in \mathbf{O}$ that $\mathbf{m} \in \mathbf{M}$.

c. *Definition 3.* (Access association on resources) An access association function **ψ: M → 2_s** returns the set of permissions belongs to or given to member **m ∊ M** at runtime.

In order to define the extended usage control operation over resources, the concept of member state, update operation, condition, and policies are introduced. The member state is defined by the set of attributes. Typically, the attributes is a string of arbitrary characteristics about members operations.

d. *Definition 4.* (Member State) A member state **t: γ(M) → com(γ(M))** is a function that maps a member's attributes to their values. This is a dictionary of attributes associated with each member. **γ** is a function that provides a set of attributes associated to a member, and a set of all possible values for an attribute a is denoted as **com(a)**. The attributes can be updated via attribute update function.

8 Bef-authorization models

Prior to access to a resource by a member in the Fabric extension, the decision function is added in the proposed model.

e. *Definition 5.* Previously stated additional usage functions the proposed model has the following components.

M,R,O, STATUS(O), STATUS(R) and befA(Member, Resources, Operation, Member Attribute, Resource Status (Attribute) and before authorization respectively);

granted (m,r,o) = befA((STATUS(m)), STATUS(r), O)

Similarly, an additional before-update is also needed, which can be defined below.

f. *Definition 6.* befUpdate(STATUS(M)), befUpdate(STATUS(R)), that is an optional operation to update on attribute of members and resources.

8.1 Ongoing-authorization models

In Ongoing-Authorization model the Fabric will request a resource usage with the absence of pre-decision making. However, continuous authorization of the resources is constantly monitored. When the model is enabled, a member that is currently allowed to use a resource will be prevented to access the resource if certain requirements are not adhered. In Hyperledger Fabric, such model is very useful when resource are accessed for a long duration of time. This functionality can be further extended into four different models shown below.

- A0 model with no **bef-updates**
- A1 model with bef-updates
- A2 model with continues-updates
- A3 model with after-updates

g. *Definition 7.* A0 model has the following components: M,R,O, STATUS(O) and STATUS(R) are not changed from Bef-authorization Model.

onA(Continues-authorization)

allowed(m,r,o) ⇒ True

stopped(m,r,o) ⇐ conA(STATUS(m),STATUS(r),O)

h. *Definition 8.* A1 model is similar to A0 model except it updates following pre-updates processes: *befUpdate(STATUS(M)), onUpdate(STATUS(r)),* and optional procedure to perform update operations on *STATUS(m)* and *STATUS(r),* respectively.

i. *Definition 9.* A2 model is also similar to A0 model except it adds following continues-update processes: *onUpdate(STATUS(m)),onUpdate(STATUS(r)),* an optional procedure to perform update operations on *STATUS(m)* and *STATUS(r),* respectively.

j. *Definition 10.* A3 model adds post-update processes to A0 model: *afterUpdate(STATUS(o)), afterUpdate(STATUS(o)),* an optional procedure to perform update operations on*STATUS(o),* respectively.

8.2 Bef-obligation models

Hyperledger Fabric introduces new obligation feature in its architecture. Using this feature, it allows a member to access a resource, if and only if, the member provide information in the authentication process. Fabric can support only for first-time authentication. On the contrary, the proposed model will check such obligation continuously.

k. *Definition 11.* Bef-obligation model has the components *M, R, O, STATUS(M),* and *STATUS(R)* which are not changed from pre-authorization model; *OBS, OBO,* and *OB*, (obligation subjects, obligation objects, and obligation actions, respectively);

befB, and *befOBL,* (bef-obligation predicates and bef-obligation elements, respectively);

befOBL ⊆ OBS x OBO x OB;

preFulfilled: OBS x OBO x OB → true, false;

getBefOBL:M x R x O → 2^{befOBL}; //a function to select bef-obligations for a requested usage

preB(m,r,o) = Λ (obs i,obo i,ob i) ∊ getBefOBL(m,r,o) preFulfilled(obs i,obo i,ob i);

preB(m,r,o) = true by definition if getBefOBL(m,r,o) = φ;

allowed(m,r,o) ⇒ preB(m,r,o).

l. *Definition 12*. The preB1 model is identical to *befBO* except it adds the following bef- update processes: *befUpdate(STATUS(M)), befUpdate(STATUS(R))*: an optional procedure to change certain attributes as a consequence of bef-obligations.

m. *Definition 13*. The preB3 model is identical to *befBO* except it adds following post-update processes: *afterUpdate(STATUS(m)), afterUpdate(STATUS(r))*:an optional procedure to change certain attributes as a consequence of bef-obligations.

8.3 Continues-obligations models

Continues obligation model is similar to bef-obligation model except it requires obligation model to be verified periodically. To achieve this model, Time parameter T is introduced as part of the continues obligation model. The time factor depends on either a time-based period or it can be an event based e.g. a member in the Fabric can be requested to submit partial information after an hour or after a specific amount of resource utilization. Based on mutability issues, continuous obligation model can be extended to four models i.e. B0 model includes continues-obligations predicate instead of bef-obligations predicate. B1, B2 and B3 are same as B0 except that the models include the bef-updates, continues-updates and after-updates, respectively.

n. *Definition 14*. The B0 model has the components *M,R,O, STATUS(O), STATUS(R), OBS, OBO,* and *OB*, which are not changed from *befB*; where T is a set of time or event elements.
onB and *onOBL*, (continues-obligations predicates and continues-obligation elements, respectively);
onOBL $\subseteq$ *OBS x OBO x OB x T;*
getOnOBL: M R O$\rightarrow 2^{OnBL}$; //a function to select continues-obligations for a requested usage
onFulfilled: OBS x OBO x OB x T $\rightarrow$ *true, false;*
onB(m,r,o) = Λ *(obs i,obo i,obi,ti)* $\in$ *getOnOBL(m,r,o) onFulfilled (obs i,obo i,ob i,ti);*
onB(m,r,o) = true by definition if getOnOBL(m,r,o) = θ;
allowed(m,r,o) $\Rightarrow$ *true;*
stopped(m,r,o) $\Leftarrow \neg$*onB(m,r,o)*

o. *Definition 15*. The B1 model is identical to B0 except it adds following bef-update processes.
befUpdate(STATUS(m)), befUpdate(STATUS(r)): an optional procedure to change certain attributes as a consequence of bef-obligations.

p. *Definition 16*. The B2 model is identical to B0 except it adds following continues-update processes.
onUpdate(STATUS(m)), onUpdate(STATUS(r)): an optional procedure to change certain attributes as a consequence of bef-obligations.

q. *Definition 17*. The B3 model is identical to B0 except it adds following post-update processes.
afterUpdate(STATUS(m)), afterUpdate(STATUS(r)): an optional procedure to change certain attributes as a consequence of bef-obligations.

8.4 Bef-conditions model

Conditions define some environmental variables through which certain policies can be enforced. e.g. an organization's employees are allowed to use its resources only when they are within the premises of organization. Such restriction is not directly associated with any resources defined in the Fabric. In a bef-condition model such conditions are checked each time when the resources are accessed. Bef-condition model introduces bef-conditions predicate (preC) that has to be evaluated before request rights are exercised.

The following definitions formalize the preC model.

r. *Definition 18*. bef-condition model has the following components:
M,R,O, STATUS(O), and *STATUS(R)* are not changed from *preA*
befCON;//a set of bef-conditions elements
getBefCON: M x R x O $\rightarrow$ *2 befCON;*
preConChecked: befCON $\rightarrow$ *true, false;*
preC(m,r,o) =Λ*preConi*$\in$*getBefCON(m,r,o)preConChecked (preConi)*
allowed(m,r,o) $\Rightarrow$ *preC(m,r,o)*

9 Continues-conditions model

In many cases, environmental restrictions have to be satisfied while rights are in active use. This could be supported by the onC model. In onC, usages are allowed without any decision process at the time of requests. However, there is a continuous condition predicate to check a particular environmental status repeatedly throughout the usages. As mentioned earlier, the onC0 model is intrinsically immutable.

The following definitions formalize the on C model.

s. *Definition 19*. The onC0 model has the following components: M,R,O, STATUS(O), and STATUS(R) are not changed from preA;
onCON;//a set of continues-conditions elements
getOnCON: M x R x O $\rightarrow$ *2 onCON;*

onConChecked: onCON → true, false;
onC(m,r,o) = onConi∈getOnCON (m,r,o)onConChecke-d(onCon i) allowed(m,r,o) ⇒ true;
stopped(m,r,o) ⇐ ¬onC(m,r,o)

Based on the study, the core Hyperledger Fabric model is extended with the above extra models. Such method will enrich future application with variety of applications based on their business needs.

10 Traditional access control adoption with ucon

The proposed DistU model can be incorporated in traditional MAC, DAC, and RBAC. Most traditional access controls and trust management can be realized by the A0 model while some extension requires the A1 model. In literature, continues-authorization and variable mutability are rarely discussed, therefore only slightly new features are available to the traditional models.

10.1 Mandatory access control (MAC)

The DistU, A0 model supports traditional MAC.

Mandatory access control operates on policy based and as well as the traditional DAC access control. The well-known targeted policy can be implemented on the new model for Fabric. The subject and objects are the members and resources. The targeted policy is defined and invoked via the A0 model. Once the policy/label is matched the permission will be granted, otherwise, the owner of the resources will not have the access.

MAC in DistU Example:

L is a lattice of security labels with dominance relation ≥

clearance: M → L
maxClearance: M → L
classification R → L
STATUS(P) = {clearance, maxClearance}
STATUS(A) = {classification}

10.2 Role-based access control (RBAC)

DistU A0 model also can support RBAC in the authorization process. In RBAC model, a role is a collection of users and operations on resources. Therefore, in the proposed model user-role assignment can be viewed as subject attributes while the permission-role assignment as attributes of object and rights. The following example shows the method in which RBAC can be viewed in the DistU models.

RBAC in DistU Example:

M = (R, O)

ROLE is a partially ordered set of roles with dominance relation ≥

actRole: M → 2 ROLE
M role: M → 2 ROLE
STATUS(M) = actRole
STATUS(R) = P role
allowed(m,r,o) ⇒ ∃role∈actRole(m), ∃role0∈M role(r, o), role ≥ role 0

10.3 Discretionary access control (DAC)

DistU A0 model also can support DAC, where policies govern the access of users to an object based on individual or group identities of users and objects. The access modes such as read, write, or execute are granted to a user if the user has the privilege to use a specific access mode on an object. In DistU A0 Model, DAC can be expressed by using either ACLs or capability lists.

DAC in DistU Example:

G is a set of groups of subject p
groupId: M → 2 G; many to many mapping
ACL: R → 2 G x R; g is authorized to do s to a
STATUS(m): {groupId};
STATUS(R): {ACL};
allowed(m,r,o) ⇒ {(g, r)| g ∈ groupId(p)} ∩ ACL(r) 6 = θ;

11 Implementation of extended permission model

Hyperledger Fabric uses chaincode or smart-contracts to execute business logic. Different members join a fabric network to share valuable private data. Chaincode updates or query the states of the resources in the distributed shared ledger. A user or a system can interact with the Hyperledger Fabric at three levels i.e. at peer level using system chaincode, chaincode level using user chaincode transactions and channel level using events stream source. All three levels are considered resources because the shared ledgers are accessed at these levels. Hyperledger Fabric allows to define the Access Control List (ACL) policy for these resources and introduces a different type of roles that are defined in each channel configuration. These roles are named as Reader, Writer, and Admin. The peers that are identified by the MSP service and assigned the Writer role can participate in the endorsement policy and can submit a transaction proposal. After endorsement, the peer updates the state of DSL, while the peer with the role reader cannot take part in endorsement policy and is unable to alter the state of the distributed ledger. However, such security can also be implemented on the user-defined chaincode level where developers are responsible to create and develop all these policy constraint. Different channel chanincode can also call each other but only in read-only mode.

Hyperledger Fabric architecture supports two types of endorsement policies, chaincode level, and key-level endorsement policy. The proposed model introduces a new system chaincode called DistU Chaincode. This new chaincode works as a reference monitor and respond against the violation of the defined resource policy. In key-level endorsement policy, a new DistU policy configuration is introduced whose key attributes set the usage limit (e.g. the number of attempts to read or write) and the conditions (e.g. IP, time, location etc.) along with the restriction level (Allow, Deny, Revoke, Alert).

DistU chaincode monitors the ongoing transaction for the mentioned key-value. When the access of a peer exceeds the limit, the monitor responds to the user chaincode with an appropriate action mentioned in the policy. With each endorsement request, DistU chaincode traverse the ledger against the required policy attributes and verify them with the requested values. Other environment attributes like IP, time and location are provided with each transaction request which are also checked against the policy. The sequence diagram in Fig. 4 demonstrates the working of the system. The end user will generate a transaction request through the invoke function of the application chaincode. That request is signed by the committer peer and sends it to all the endorser nodes that are defined in the core configuration file of the channel.

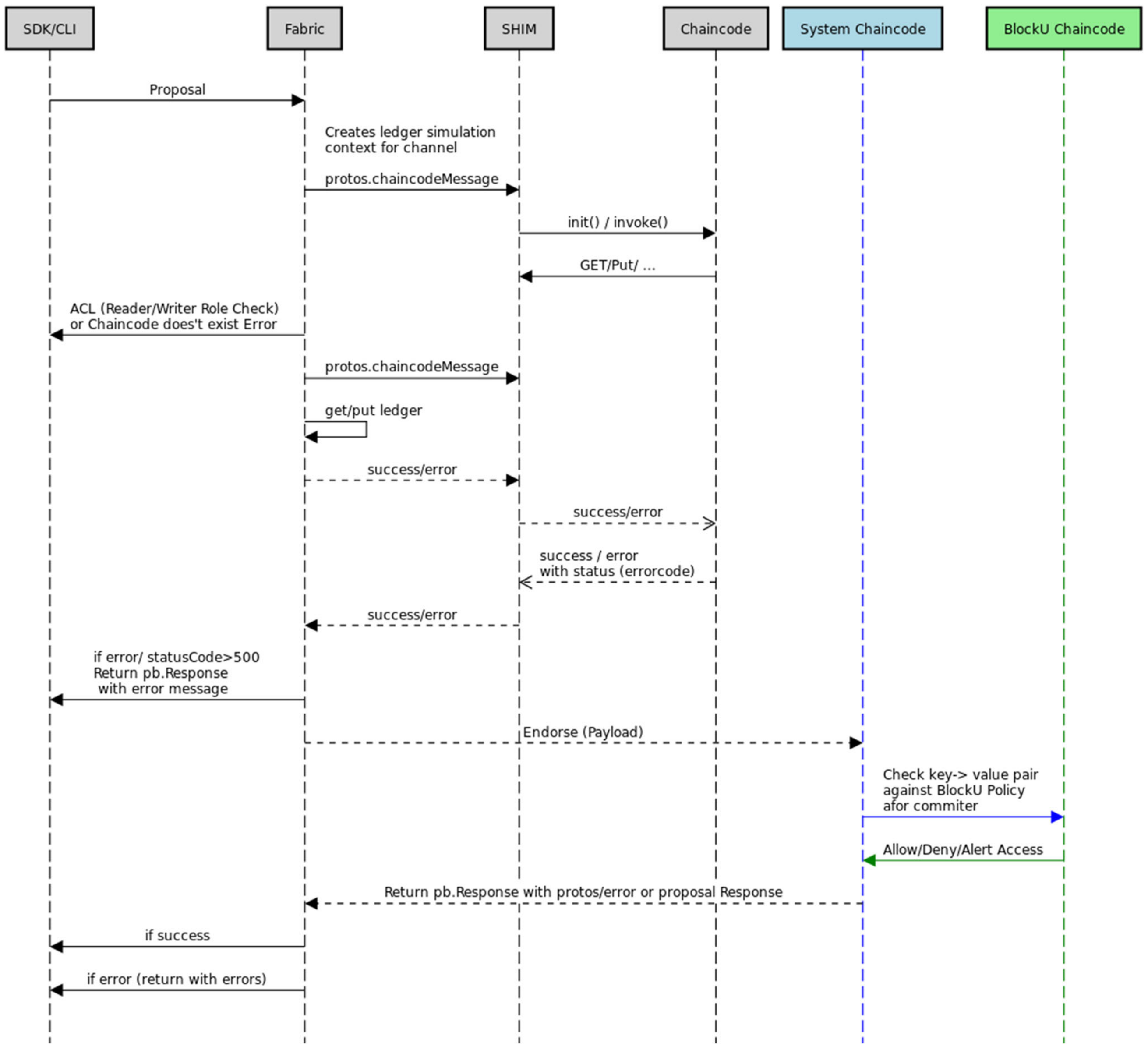
SDK/CLI
Fabric
SHIM
Chaincode
System Chaincode
BlockU Chaincode
Proposal
Creates ledger simulation context for channel
protos.chaincodeMessage
init() / invoke()
GET/Put/ ...
ACL (Reader/Writer Role Check) or Chaincode does't exist Error
protos.chaincodeMessage
get/put ledger
success/error
success/error
success / error with status (errorcode)
success/error
if error/ statusCode>500 Return pb.Response with error message
Endorse (Payload)
Check key-> value pair against BlockU Policy afor commiter
Allow/Deny/Alert Access
Return pb.Response with protos/error or proposal Response
if success
if error (return with errors)

Fig. 4 DistU model sequence diagram

Endorser receives the request and calls for the system chaincode to check it against the ACL (check if the role of the committer reader, writer or admin) and signature key. Transaction proposal has all the attributes like chaincode arguments, read set and write set values of the resource (key-value) are verified. The proposed DistU chaincode plugin is also initiated at this point and locates the key level enforcement policy. Whenever the DistU chaincode finds a key value, it traverses the ledger against it. The traversed data is compared with the requested transaction. Once the policies are rigorously verified it returns the response to the user chaincode. Based on the defined policy the DistU can accept or deny a transaction proposal.

11.1 Use case

A detailed permission model is described that enables the members of a business network and the owners of the resources to specify usage-based constraints while performing a transaction. The new permission model is tailored explicitly for Hyperledger Fabric.

As previously stated, attribute mutability has three states in the UCON engine, which are pre-update, on-update, and post-update. Bef-update attributes are set before accessing the resource. In a particular use-case example, the start time is initially set to 0 and after 24 h the resource will be revoked. Once the resource is moved to the accessing state, the mutable attribute will update the time and warnings accordingly. The warnings will be set to 0 to ensure that once the number of warnings reaches the allotted time for the resource, the access to the resource will be revoked. Consequently, each state change of UCON will be stored in transparent ledger, in case of a dispute or if the owner would like to verify for further deduction from the renter. In contrast to attribute states, the second condition will be constantly monitoring the defined zone, such as the distance in which a rented vehicle may travel. The current location coordinates of the car will be defined as a pre-update attribute and that will be used to constantly monitor the car location. The warning attribute will be set to 0 along the location. As soon as the car location is identified as beyond the designated range or if the user receives 5 warnings, access to the car will be revoked unless the owner maintains contact with the renter. In such case, all those transactions will also be locked in the trusted network peers (Fig. 4).

12 Conclusion

A number of distributed solutions are provided for the establishment of trust and imposing transparency. However, these solutions have problems associated with them. This research study provides a comprehensive study on the various mechanism for access control mechanism and extends the permissioned blockchain access control model that can perform continuous monitoring of the object over the member's acess. Initially, the formal model of the extended permissions is discussed, and finally, the Hyperledger Fabric is selected to incorporate the newly designed formal access control model. There are many business use-cases that may not be feasible to be implemented on the basic role-based access control. However, it is deemed that usage control model is always difficult to implement. In fact, this is also necessary for the variety of business application in the future and even the existing one requires this extended permission model.

Funding This research work is a collaboration of Universiti of Kuala Lumpur (UniKL/CoRI/str15101) and Islamic University of Madinah. Special thanks to Deanship of research of its support in every aspect.

References

1. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved from https://bitcoin.org/bitcoin.pdf.
2. Fabric, H. (2018). Key concepts: Identitty. 2018. https://hyperledger-fabric.readthedocs.io/en/latest/identity/identity.html. Visited on 10/06/2018. Cit. on p. 25.
3. Vukolić, M. (2016). The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In Lecture Notes in Computer Science (LNCS) (Vol. 9591, pp. 112–125). Berlin: Springer.
4. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., & De Caro, A. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In *EuroSys '18 proceedings of the thirteenth eurosys conference*. Porto, Portugal: ACM.
5. Dhillon, V., Metcalf, D., & Hooper, M. (2017). The hyperledger project. In *Blockchain enabled applications* (pp. 139–149). Berlin: Springer.
6. Ali, T. (2018). Z notation formalization of blockchain healthcare document sharing based on CRBAC. *Journal of Information Communication Technologies and Robotics Applications (JICTRA), 9*, 16–29. Retrieved from http://nicerjcs.com/index.php/cs/article/view/179
7. Xu, R., Chen, Y., Blasch, E., & Chen, G. (2018). BlendCAC: A blockchain-enabled decentralized capability-based access control for IoTs. *Computers, 7*(3), 39. https://doi.org/10.3390/computers7030039.
8. Xu, R., Chen, Y., Blasch, E., & Chen, G. (n.d.). A federated capability-based access control mechanism for internet of things (IoTs). Paper presented at SPIE defense and commercial sensing 2018 (DCS) conference, Florida, USA.
9. Cruz, J. P. (2018). RBAC-SC: Role-based access control using smart contract. *IEEE Access, 6,* 12240–12251. https://doi.org/10.1109/ACCESS.2018.2812844.
10. Outchakoucht, A., Hamza, E.-S., & Leroy, J. P. (2017). Dynamic access control policy based on blockchain and machine learning for the internet of things. *International Journal of Advanced Computer Science and Applications, 8*(7), 417–424. https://doi.org/10.14569/issn.2156-5570.

11. Lindqvist, H. (2006). Mandatory access control. (Unpublished master's dissertation). Umea University, Department of Computing Science. Sweden.
12. Fabric, H. (2018). Hyperledger fabric ledgers. 2018. https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html?highlight=LEDGER. Visited on 10/06/2018. Cit. on pp. 31–33, 48.
13. Fabric, H. (2018). Key concepts: Membership. 2018. https://hyperledger-fabric.readthedocs.io/en/latest/membership/membership.html. Visited on 10/06/2018. cit. on pp. 25–27, 30.
14. Ali, J., Ali, T., Musa, S., & Zahrani, A. (2018). Towards secure IoT communication with smart contracts in a blockchain infrastructure. *International Journal of Advanced Computer Science and Applications (IJACSA)*. https://doi.org/10.14569/IJACSA.2018.091070.
15. Park, J., & Sandhu, R. (2004). The UCON ABC usage control model. *ACM Transactions on Information and System Security (TISSEC), 7*(1), 128–174.
16. Zhang, X., Parisi-Presicce, F., Sandhu, R., & Park, J. (2005). Formal model and policy specification of usage control. *ACM Transactions on Information System Security, 8*(4), 351–387.

Muhammad Yasar Khan received the M.S. degree in Computer Sciences from the Institute of Management Sciences, Peshawar, Pakistan, in 2016. After completion of his Master Degree in Computer Sciences in 2008, he joined Security Engineering Research Group, Institute of Management Sciences, Peshawar in 2009 as Android System and Architecture Developer, where he worked on many areas of Security Engineering such as Integrity Measurement Architecture, Android Permission Extension, Usage Control Engine and on Cross Domain Access Control Management. He is currently doing his Ph.D. in Information Technology at Universiti of Kuala Lumpur, Malaysia. His current research interests include Blockchain, IoT, malware analysis, machine learning and Cross Domain Access Control Management.

Megat F. Zuhairi is a Senior Lecturer within the System and Network Section in Malaysian Institute of Information Technology, Universiti Kuala Lumpur. He received his Ph.D. in Electronics and Electrical Engineering from the University of Strathclyde in 2012 and M.Sc. in Communication Networks and Software from the University of Surrey, UK in 2002. He is currently an active researcher and a certified Cisco Network Academy Instructor at the institute. His research interests include computer data networking, wireless mobile ad hoc communications, and Blockchain.

Toqeer Ali is an Assistant Professor at Islamic University of Madinah. He received his Ph.D. in Information Technology in 2014 from Universiti Kuala Lumpur, Malaysian Institute of Information Technology. His research interests are Blockchain, System Security, Operating System, Deep Learning.

Turki Alghamdi is currently working as an Assistant Professor, the Dean, and the Founder of he Faculty of Computer and Information Systems at Islamic University in Madinah, KSA. He received a B.Sc. in Computer Science from Taif University, KSA in 2005, and M.Sc. in Software Engineering from University of Bradford, UK in 2008. He received a Ph.D. in Software Engineering from De Montfort University, UK in 2012.

Jose Antonio Marmolejo-Saucedo is a Professor at Panamerican University, Mexico. His research is on operations research, larges-scale optimisation techniques, computational techniques and analytical methods for planning, operations, and control of electric energy and logistic systems. He received his Doctorate in Operations Research (Hons) at National Autonomous University of Mexico. At present, he has the second highest country-wide distinction granted by the Mexican National System of Research Scientist for scientific merit (SNI Fellow, Level 2). He is a member of the Network for Decision Support and Intelligent Optimisation of Complex and Large Scale Systems, Mexican Society for Operations Research and System Dynamics Society. He has research articles in science citation index journals, books, conference proceedings, presentations and book chapters.