IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 April 2024, accepted 13 April 2024, date of publication 18 April 2024, date of current version 26 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3390842

RESEARCH ARTICLE

A Blockchain Based Scalable Domain Access Control Framework for Industrial Internet of Things

MUHAMMAD USMAN[1], MUHAMMAD SHAHZAD SARFRAZ[1], (Senior Member, IEEE), MUHAMMAD UMAR AFTAB[1], USMAN HABIB[2], AND SALEHA JAVED[3]
[1]Department of Computer Science, National University of Computer and Emerging Sciences, Chiniot-Faisalabad Campus, Chiniot, Islamabad 35400, Pakistan
[2]Software Engineering Department, FAST School of Computing, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan
[3]Machine Learning Group, SRT, Luleå Technical University, 971 87 Luleå, Sweden

Corresponding author: Muhammad Shahzad Sarfraz (shahzad.sarfraz@nu.edu.pk)

ABSTRACT Industrial Internet of Things (IIoT) applications consist of resource constrained interconnected devices that make them vulnerable to data leak and integrity violation challenges. The mobility, dynamism, and complex structure of the network further make this issue more challenging. To control the information flow in such environments, access control is critical to make collaboration and communication safe. To deal with these challenges, recent studies employ attribute-based access control on top of blockchain technology. However, the attribute-based access control frameworks suffer due to high computational overhead. In this paper, we propose an improved role-based access control framework using hyperledger blockchain to deal with IIoT requirements with less computational overhead making the information control process more efficient and real-time. The proposed framework leverages a layered architecture of chaincodes to implement the improved access control framework that handles the permission delegation and conflict management to deal with the dynamism of the IIoT network. The system uses a Policy Contract, Device Contract, and Access Contract to manage the workflow of the whole access control process. Each chaincode in the proposed framework is isolated in terms of its responsibilities to make the design low coupled. The integration of improved access control with blockchain enables the proposed framework to provide a highly scalable solution, tamper-proof, and flexible to manage conflicting scenarios. The proposed system outperforms the recent studies significantly in computational overhead in extensive simulation results. To verify the scalability and efficiency, the proposed is evaluated against a large number of concurrent virtual clients in simulation and statistical analysis proves that the proposed system is promising for further research in this domain.

INDEX TERMS Blockchain, smart contract, IIoT, access control, conflict management, policy contract, access contract, device contract, hyperledger fabric.

I. INTRODUCTION

The Internet of Things (IoT) [1] is widely used to refer to the network where different objects are interconnected for business applications without human intervention using a network medium within a digitized environment. The interconnected devices range over different types from homogeneous to heterogeneous devices such as sensors, actuators, smart vehicles, house appliances, wearable devices, etc [2]. These devices can sense, capture information, and process it for better decision-making in an automated fashion. IoT-based networks are proven to be enabling technology that can transform many industries including agriculture, healthcare, transportation, and the industrial realm, which stand to benefit greatly from the IoT and its supporting technologies [3]. One such application of an IoT-based network is the IIoT [4] where smart machines are interconnected to automate industrial processes for enhanced efficiency

The associate editor coordinating the review of this manuscript and approving it for publication was Stefano Scanzio.

and productivity. The emergence of IIoT has transformed industrial businesses and enabled them to achieve business goals with less effort. The applications of IIoT include Cyber-Physical Systems (CPS), Machine-to-machine (M2M) data exchange and translation, and many more [5]. Such applications of IIoT are referred to as Industry 5.0 [6], [7].

The data exchange between the devices can be achieved using lightweight protocols such as MQTT, Zigbee, CoAP, DDS, LoRa, and many others with each protocol having its strengths [8]. This way IIoT networks provide an enabling platform to build automated solutions for various domains of industries. However, the complete application of IIoT involves the integration of other technologies using network gateways such as cloud computing, edge computing, and blockchain for storing data. The data storage can provide analytics and remote access to various stakeholders of industry businesses for enhanced decision-making [9]. The intuition behind IIoT is that smart devices are better at capturing and analyzing data in real-time, along with communicating important information that can be used to infer and predict business decisions quickly and accurately. For business intelligence efforts, the sensors and actuators can be used to capture the inefficiencies and problems quickly as compared to existing traditional systems thus saving more time and money [10]. The applications of IIoT include automatic quality control, sustainable and green practices, supply chain traceability, and overall supply chain efficiencies [11].

However, with great potential to offer, the IIoT network suffers from scarcity challenges as interconnected devices are constrained in terms of resources and processing power [12]. Similarly, the limited bandwidth and latency can further enhance this issue. Further, the limitations of the network become more challenging due to any adversarial request to the network resources or a node becoming part of the network illegally [13]. To restrict such adversarial nodes many researchers have explored the integration of access control (ACL). There are several types of ACL frameworks however, role-based access control (RBAC), attribute-based access control (ABAC) and hybrid frameworks are more popular [13]. ACL for such scenarios becomes promising due to its direct application to restrict the network resources only to validated nodes. It helps the network to determine the access privileges of devices based on the device role or contribution it provides in the whole network [13]. A complete ACL framework provides the formulation of access policies that determine the different levels of access on the device. The policies help the ACL framework to deal with information with levels of privacy for network resources restricting the access of unauthorized devices or users [13].

Considering the dynamic nature of IIoT networks where the network size and structure can be changed rapidly, the traditional ACL frameworks are not suitable to plug in with such networks [14]. These networks involve heterogeneous devices that share different types of data with different privacy levels. Similarly, the data exchange involves the information transmission using gateways for further processing and storing to third party components. Such issues require the ACL to at least provide the privacy of the information, adaptable policies to handle the unseen and conflicting scenarios, and agentability of the access control [14]. Many researchers have proposed ABAC solutions for this purpose but these systems based on the ABAC suffer due to the high computational overhead of ABAC [15]. In comparison to ABAC systems, RBAC is quite simple in the way it works but it fails to deal with IIoT applications due to the complex nature of the network. In RBAC systems, conflicting scenarios can arise in the network where many devices with different roles can request the same resource for different reasons [13]. For example, there can be good chances of devices being changed in their contribution to the network forcing a change in the access policies. Thus these rapid changes in the access policies can lead to conflict with the other devices available in the network. This makes integrating traditional ACL in IIoT more difficult and needs appropriate tweaks to support the networks better. Similarly, the ACL for IIoT requires a highly scalable and robust to properly accommodate the new nodes being part of the network for extension of the network [13].

Many researchers attempted to provide tweaked solutions to address this problem to make ACL better support the requirements of the IIoT [14]. However, the conflicting roles and adaptable policies for handling the conflicting permissions in such scenarios is still an open problem [16]. Recent studies have employed the blockchain to address the distributed nature of the devices in the network and proposed flexible ACL frameworks [17]. The applications for blockchain for IIoT have been a popular topic among researchers due to the benefits it provides. Further, many researchers have also explored the use case of different blockchain types such as permissionless, permissioned, and consortium for various IIoT application scenarios [18], [19]

However, in the context of ACL, very little or no considerations are provided on permission delegation for conflicting scenarios in such networks. Further, the permission delegation can be borrowed using the principle of least privileges for such systems to make it more scalable. Similarly, as the blockchain provides the decentralization itself, a better sense of scalability can be achieved when permission delegation and revocation are implemented on the top of blockchain [20]. Despite this, the recent studies also fail to provide a controllable framework where the distribution of access privileges and their exchange between the requesting devices can be confirmed or audited.

In this paper, we propose a permission-constrained RBAC model empowered with smart contracts to resolve the dynamic policy conflict while maintaining high scalability. Smart contracts are programmable components by which business logic can be implemented in the blockchain network. The proposed framework implements the scalable ACL by leveraging the hyper ledger fabric blockchain for industrial settings to address these problems. The system established a dynamic policy where new devices can be joined or removed simultaneously. Similarly, the proposed system allows the devices to have multiple mutually exclusive roles. For the proposed system, the conflicting roles in RBAC are considered to resolve them in real-time for helping the IIoT networks against security breaches. A device can have mutually exclusive roles that can conflict with each other but the device cannot act in both roles during the same session. The resolution mechanism uses permission delegation and revocation in case of policy conflict to avoid the deadlock situation with no significant effect on the efficiency of the network. The policy adaptation in such scenarios considers the least required resources by conflicting roles and updates the ACL policy. The system state can be restored afterward in an iterative fashion after the conflict is avoided and a new request for the same resource is put by devices. The network entertains the new request if no further conflict is detected. To make the framework open, dynamic, adaptive, and scalable we use the policy domain based on the smart contracts where each device is distributed among these policy domains. For ACL management, where authority is required for the permission delegation, revocation, and consensus between the devices is needed, the proposed system introduces the agent in the smart contract architecture for decision-making in critical scenarios such as conflict arising either in roles or policy. Further, the overall ACL management is auditable, traceable, and policy domain interactions are secure inherently by blockchain. The proposed system is feasible to apply in any industrial process where interconnected devices require trust management with information exchange while maintaining privacy at different levels. The main contributions of this paper are as follows:

1) A lightweight permission-constrained access control framework is proposed using RBAC to conform with IIoT network requirements. The proposed system is novel in this domain and handles the conflicting roles. The conflicts are resolved with permission delegation and revocation in such scenarios making it more robust for such distributed networks.
2) The proposed system is scalable and prone to network dynamics in complex scenarios by using the adaptable ACL policies. The framework can dynamically update its state to network topology changes where new devices are added or removed from the IIoT network.
3) The proposed framework leverages the potential of blockchain by integrating an open source hyper ledger blockchain with our ACL model. The integration of blockchain helps the system to support the distributed characteristics of the network dynamics and provides remote access to different stakeholders of the system.
4) A multi-domain policy representation is presented with smart contract segregation that makes the system low-coupled and highly cohesive. The policy domains are segregated based on assigned privileges over the roles. This also makes the ACL model auditable and traceable for future ACL management.
5) With evidence of simulation results, the proposed system is efficient and highly performing due to better ACL management with agents available in the blockchain. The agents are responsible for the policy updation, and approvals for the authority delegation making the overall system automated. This automation in decision-making enables the model to achieve high performance with respect to time for operations over the blockchain.
6) A complete prototype of blockchain implementation is presented with comprehensive results and comparison to baseline studies. The proposed model demonstrates a significant performance boost in comparison to baseline studies representing its robustness for ACL operations.

The rest of the paper organization is as follows:

A brief overview of incorporated technologies is provided in the background section followed by the related work section. The related section discusses the recent state-of-art studies and needs of the proposed system followed by the proposed model section. In the proposed model section, a complete conceptualization of the model is presented with a system model, blockchain smart contracts segregation into domain levels, process activity, and flow of the information. After the proposed model, the results section presents a complete comparison with baseline studies in terms of different quantitative and qualitative attributes followed by the limitations and conclusion section.

II. BACKGROUND OVERVIEW

The emergence of IIoT is new and promising due to vast applications in many business domains making the processes automated and scalable. However, these applications need to be mature for handling the authentication and validation of the smart entities participating in the network to protect the public and private information of the network [21]. Any breach of this information regardless of the privacy level, can directly impact businesses. This section provides a brief overview of underlying technologies that help us implement a novel approach for lightweight and scalable ACL models for such networks to protect the information flow.

A. ACCESS CONTROL

Access control is a process to protect the resources from invalid usage in the network. The model specifies the objects as resources of the network for which subjects can request multiple operations in the network [13]. The subjects can be a user, entity, device, or service that can participate in the network. Several ACL methods are proposed in the research paradigm including discretionary ACL (DAC), RBAC [22], ABAC [23], and a mixture of these models to make hybrid ACL systems [13]. With each ACL system having its strengths, its usage will guarantee the protected usage of the network resources especially in distributed environments.

In this paper, the strength of the RBAC is focused on making it more flexible for the IIoT environment. As the IIoT environment is automated, dynamic, and complex, the change in the device state can create conflicting scenarios in the network. The main focus of this paper is to make RBAC flexible to resolve such issues concerning the requirements of the IIoT requirements to support better scalability without any additional cost overhead. Figure 1 shows the naive representation of the RBAC system in working.

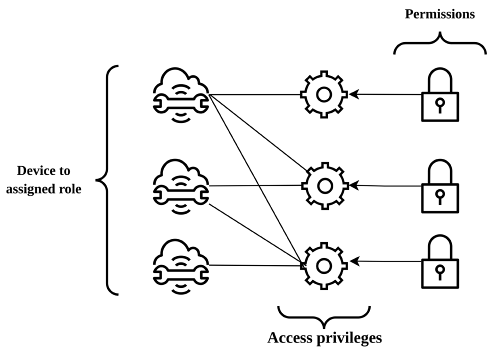


FIGURE 1. **A naive working of RBAC system.**

B. BLOCKCHAIN

The IIoT is heavily dependant on middleware technologies such as cloud computing, and edge computing. The key reason behind this dependence is that IIoT is a network of resource-constrained devices [24]. These devices have limited capabilities in terms of data storage, processing, and smart decisions. To make IIoT efficient, cloud computing and edge computing were explored by researchers to scale these networks for various business domains [24]. However, with these middleware technologies integrated into the IIoT network, there were still key problems such as privacy, proof-of-identity, and unauthorized physical access to end devices [25]. With the emergence of blockchain, these issues were attempted for an efficient solution as blockchain provided a way to establish trust, the privacy of information at different levels, and tamper-proof transactions for information exchange between these interconnected devices [25].

Blockchain refers to the decentralized storage of data into several batches called blocks. Each block is linked with other blocks in chronological order making a chain in the network. The chain of blocks is broadly referred to as blockchain. However, blockchain consists of several key components such as a decentralized ledger, consensus algorithm, smart contracts, and cryptographic hashing algorithm. The ledger is stored on multiple nodes in the network and each update in the state is validated using the consensus algorithm [25]. The consensus algorithm ensures that each node in the blockchain agrees to the current state of the network. Further, using the cryptographic hash, each block also contains the hash of the previous connected block in the ledger making it nearly impossible to tamper and alter.

In the existing literature, there exist different types of blockchain frameworks that provide different working models of decentralization at the core [26]. These frameworks are majorly categorized into public and private networks. The public networks are open to anyone and data is available broadly for read and write. The public networks are usually suitable for decentralized financial use cases such as Bitcoin and Ethereum [26]. However, private networks require predefined rules for becoming a member of a blockchain network by providing proof of identity. These networks are largely categorized into permissioned and consortium networks.

C. HYPERLEDGER FABRIC

Hyperledger fabric (HF) is an open-source blockchain platform that supports the modular and distributed approach that suits the requirements of the IIoT networks [27]. The HF not only supports the decentralized ledger, group consensus, and immutability but also supports many consensus approaches such as PBFT, RAFT, Solo, and Kafka [28]. With this efficient consensus mechanism, the network can enable the model to achieve higher throughput for the ACL operations over the blockchain network. Further, public blockchain networks suffer from low transaction throughput, high latency due to proof of work (PoW), consistency issues, and the long time required for transaction confirmation. [29] This makes the HF a popular choice for private networks to use for such scenarios to support the ACL operations occurring in real time [30].

HF uses the modular approach for smart contracts, data storage, and other services in isolation from each other in the form of docker containers. The containers make the sandbox environment separating the physical objects from the application program ensuring the high security of the application. [29]. The main components of the HF include Certificate authority that generates or removes the security credentials of the participating nodes in the network. HF includes two layers of the clients where the first layer is responsible for the peer node interaction and management while the second layer serves the configurations or management of the chaincode in the network. A description of HF components is as follows:

1) Peer Nodes: The peer node is a full node in the hyper ledger and is responsible for executing the chaincodes. The execution of chaincodes is then validated by as well using the transaction signatures. After the transaction is validated, it is then included in the ledger. Based on these functionalities, the peer node is categorized into types, a) endorsing peers, and b) committing peers. The endorsing peers are responsible for validation while committing peers are responsible for including the transaction in the ledger after endorsement is achieved.

2) Orderer Nodes: Orderer is another full node in the blockchain that is responsible for achieving the consensus in the hyper ledger. However, when a transaction is endorsed by the endorsing peers, the orderer will package it into different blocks before appending it into the ledger. The blocks are then sorted into certain ordering and then sent back to committing nodes to append them to the ledger. These peers and orderer nodes work collaboratively to reach a consensus in the blockchain network.
3) Channel: The channel isolates the data in the blockchain to enhance data privacy and confidentiality. Each organization in HF can be isolated in a separate channel that can have its ledger making it multi multi-ledger, multi-blockchain network. This is an important concept that we use for our proposed method. The domains can reside in their channels securely and can interact with each other for operations over the network.
4) Chaincode: The smart contracts in HF are called chaincode in HF. The chaincode is responsible for generating transactions in the network and helps the network to manage the transactions that able the outer world to interact with HF. In our proposed model, we implement multi-domain policies using chaincode to enforce the ACL mechanism for the IIoT network.

Figure 2 represents the architecture of HF in working.

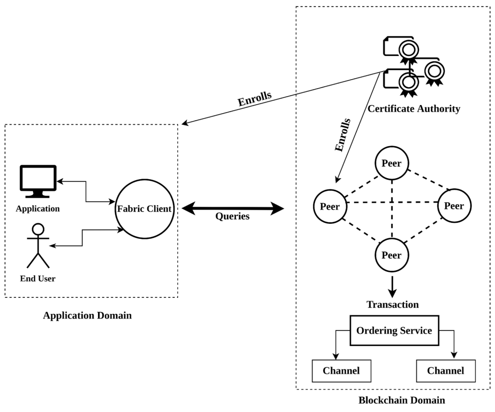


FIGURE 2. **A typical architecture of the Hyperledger Fabric in working.**

III. RELATED WORK

IoT enables the organization to connect things for information exchange. Due to the heterogeneity of the connected devices, mobility in the network makes the IoT a dynamic and complex environment [30]. Similarly, the information exchange in the network needs to be authenticated and validated for obvious reasons. For this authors in [30] proposed a private blockchain system for dynamic ACL using ABAC. The proposed system leverages the HF for the blockchain and uses the chaincode in a layered architecture. Similarly, the authors in [31] proposed a private blockchain using HF network solution for trust among the devices. The proposed model is a two-step process for achieving that with ABAC. Blockchain has been proven a promising technology for IIoT networks, however, [32] argued that the public blockchain suffers from high latency due to poor consensus mechanism. This can impact the performance of the network efficiency in real-time where information should be authenticated in the network. To resolve this issue, [32] leverages the consortium blockchain using layered architecture for the chaincodes. The solution [32] uses the three chaincodes named access control chaincode, policy management chaincode, and credit evaluation chaincode. Each chaincode is assigned mutually exclusive responsibilities for low-coupled design. The same problem is also discussed by authors in [17] where they proposed a private blockchain solution for better latency in access management issues. The proposed system leverages the RBAC for the implementation of a hybrid ACL system that provides transparency, and robustness in the existing RBAC system. The proposed system devised the three levels of network access management policies. The levels include role-based policy, rule-based policy, and organization-based policy.

Authors in [33] discussed that IIoT networks are more vulnerable to malicious attacks of identity stealing, and failing to provide reliable sources. For this problem, they propose an ABAC using a private blockchain with HF. The proposed system also uses the layered architecture for the smart contracts where each smart contract is responsible for separate tasks. The chaincodes are named device contract, policy contract, and access contract managing their work domain over the blockchain.

The integration of ACL with blockchain provides a sense of authentication for the participating nodes in the network. However, the security and immutability are provided by the blockchain as a core feature. Authors in [34] argued that blockchain systems are highly scalable and provide plug-and-play integration to third-party applications as well. this idea is used by the authors to integrate the OAuth 2.0 on the top of the RBAC system to provide improved security and privacy for the information in the private blockchain network that is implemented on HF. The proposed scheme demonstrates good performance for identity and access management in simulation results. However, the idea of managing the scalability required for IoT use cases can be done by tweaking the ACL system to deal with complex scenarios. The authors in [35] proposed a cross-domain ACL scheme to provide a robust solution for dynamic situations. The proposed scheme utilizes the consortium public blockchain to implement the lightweight ACL scheme that manages the data sharing in the network. To secure the data sharing in an improved fashion, the authors utilize cryptographic algorithms.

However, to make the computation overhead less in the system, RBAC systems are better in comparison to the ABAC systems. The RBAC demonstrates a better consensus convergence due to its simpler architecture than the ABAC [36]. The authors in [36] have utilized RBAC with public blockchain to manage the user-role permissions

TABLE 1. Comparative analysis of proposed methodology with related studies.

Sr	Paper	Access Control	Blockchain	Scalability	Permission Delegation	Controllability	Conflicting Roles
1	[30]	ABAC	HF	✓	✗	✗	—
2	[31]	ABAC	HF	✓	✗	✗	—
3	[32]	ABAC	Consortium	✓	✗	✗	—
4	[17]	Hybrid	HF	✓	✗	✗	✗
5	[33]	ABAC	HF	✓	✗	✗	—
6	[34]	RBAC	HF	✓	✗	✗	✗
7	[35]	RBAC	HF	✓	✗	✗	✗
8	[36]	RBAC	HF	✓	✓	✗	✗
9	[37]	RBAC	HF	✓	✗	✗	✓
10	[38]	ABAC	HF	✓	✓	✓	—
11	[39]	ABAC	HF	✓	✓	✓	—
10	Proposed	RBAC	HF	✓	✓	✓	✓

for a single organization. The organizational resources are protected using the RBAC against the man-in-the-middle attack as demonstrated in simulation results. Utilizing the RBAC for access management in these networks can lead to various issues. The direct implementation of RBAC fails to provide flexibility and scalability in the network. For this purpose authors in [37] proposed a flexible RBAC system based on the private blockchain using HF. The authors implemented the RBAC to detect the policy's conflicting states based on the system's rules. The conflicting states can lead the system to impact the fairness of resource access in the network. To do so, the chaincodes are responsible for enforcing the rules that check the network state against the policy rule for the possible conflicting states in the network. However, the proposed system resolves the conflicting states statically with higher latency. However, resolving the conflict is important but while resolving the conflict, the conflicting roles should not be isolated in the network. An efficient way to do this can be to remove the permissions that may cause the conflicting situation in the network. This approach makes sure that all roles stay active in the network doing their operation while the network also avoids conflict in the network. Authors in [38] proposed a permissioned blockchain for the IIoT environment for radio frequency identification (RFID) systems that perform registration, authentication, and auditing in the network. However, the policies in the proposed system are run on the RFID middleware. The proposed scheme does not involve conflicting roles rather it assigns the required permission only for the operations by the objects. However, due to changes in the network can further arise this issue. Many research studies are exploring the applications of blockchain to make IIoT largely scalable and exchange data while maintaining different privacy levels. However, there is little attention provided to deal with the issues of RBAC to further enhance it for complex and dynamic networks. Enhancing RBAC for adaptive policies and conflict management is still an open problem in existing research. Table 1 provides a technical summary of each study based on the type of network, access control process, and features it provides. Meanwhile, Table 2 compares RBAC studies based on the features these studies provide to implement the access control integrated with blockchain.

TABLE 2. A comparison of RBAC studies providing the operations in Access management.

Study	Scalability	Delegation	Controllability	Conflicting Roles
[34]	✓	✗	✗	✗
[35]	✓	✗	✗	✗
[36]	✓	✓	✗	✗
[37]	✓	✗	✗	✓
Proposed	✓	✓	✓	✓

IV. PROPOSED SYSTEM MODEL

The proposed solution is based on the private permissioned blockchain that uses the enhanced RBAC for ACL management supporting the dynamic and complex scenarios. The improved RBAC would help to deal with the rapid changes in the IIoT network. The integration of ACL operation is done using the HF blockchain. The chaincodes are implemented for the ACL management to make the proposed system provide authorization, permissioned access to resources, controllability, permission delegation, and conflicting management. The system model is discussed in detail in a later section. In section IV-A a discussion is provided on the application of the proposed model in IIoT scenarios. Section IV-E provides detailed information about the improvements done in traditional RBAC to deal with the dynamic scenarios discussed in IV-A. Finally, section IV-D discusses the integration of an improved RBAC system with IIoT application using the HF blockchain.

A. SYSTEM MODEL

The proposed model is based on a framework proposed by [39] that combines the ABAC process with the blockchain to make it suitable for the distributed nodes available in the network. We take ideas for the proposed model based on [39] using RBAC to resolve the large overhead issues by ABAC. The efficiency in computational overhead can be achieved using the RBAC but RBAC traditionally fails to comply with the changing situations in the IIoT network. The RBAC systems can better comply with the requirement if they deal with the roles in dynamic situations. This phenomenon is taken as motivation for the proposed model using RBAC and the network is defined including the following components:

Definition 1 (Role): Role is defined as the context state of any node that exists in the network. The representation of any device in this context is helpful to assign access privileges for a single or set of nodes to operate over subjects. This can be written mathematically as:

$$n_i = r_i \tag{1}$$

where $n_i \in N$, and N is a set of nodes in the network, Where $r_i \in R$, and R is a set of roles that exist in the AC policy.

Definition 2 (Policy): Policy is defined as a rule that determines the level of access privilege, or a guideline, under which a device or node is permitted for a set of actions on the resources in the network. The policy will define a constraint satisfaction that a node must meet to perform any operation. Mathematically, a policy can be defined as:

$$P(r) = (s, \hat{p}) \tag{2}$$

where r is an arbitrary role representing the context of any node in the network, s represents the subject or resource in the network, and $\hat{p}$ is a set of actions allowed for the role to perform on the resource. For example in the IIoT environmental monitoring system, where logger is an arbitrary role that can read and write into data, a policy represented in 2 can be rewritten for this as:

$$P(logger) = (data, \{Read, Write\}) \tag{3}$$

Definition 3 (Client and Service): Client is defined as a node that can further request a resource in the network based on any change in the device context to operate smoothly in the network. Considering the dynamic topology, where mobility exists in the IIoT applications, a node may initiate a request to access the additional resources to perform the assigned task. Similarly, extending this concept, we define service as a node that may provide a resource facility in the network. However, considering the autonomy in the network, where there is no central entity to manage the role, all nodes are equal in the network but these nodes must conform to a balance in the network to perform the task. The balance in the network is maintained by the agent nodes to avoid any situation that violates the maintained balance. However, clients and services are distinguished from roles due to reason that they represent the policy state and at any time, any policy in the network would be either in the client domain or the service domain.

Definition 4 (Agent): Agent is defined as a node that monitors the client and service nodes to maintain the balance in the network. The agent nodes are globally trusted nodes and are responsible for policy management, especially for detecting the conflicts in the policies.

B. INFRASTRUCTURE MODEL

For the proposed model, the distributed IIoT equipment in the physical world is integrated with decentralized and permissioned blockchain using HF. The blended integration of IIoT devices with blockchain is summarized into the following components that when combined make the whole infrastructure for the IIoT application. Figure 3 shows the layered architecture of the infrastructural model of the proposed system. All these components' interactions are shown in Figure 4. A brief overview of infrastructural components is as follows:

IIoT Nodes: The nodes are equipment that capture, process, or exchange information with each other in the network. The nodes in the information exchange process can be referred to as client and service depending on their role in the network. The nature of the node can be both physical and digital.

IIoT Users: IIoT user is referred to as an application user who provides the policies for the network initially. The users are connected with the blockchain using the interface for policy feed suitable for the network. Similarly, users can also use the network data for better decision making about the environment.

Gateway: The gateway is the hub of the network where all nodes are connected with the network for the information flow.

Blockchain: Blockchain is a communication network that provides business logic and enforces the policies with their privacy levels.

Chaincodes: Chaincodes are responsible for enforcing the overall management of the proposed system. The policies and their management are executed on the blockchain using multiple domain chaincodes.

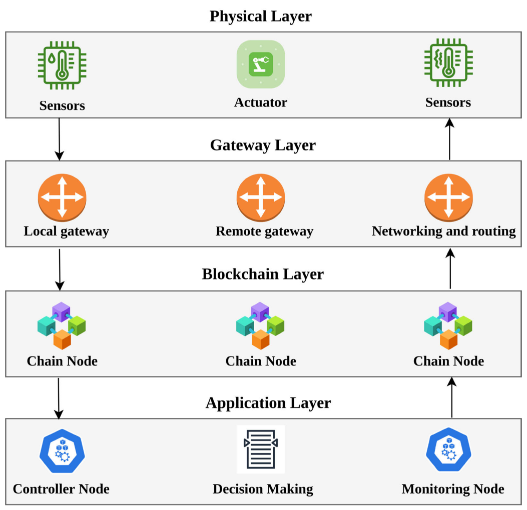


FIGURE 3. The layered representation of the infrastructure model of the proposed framework.

C. IIOT APPLICATION SCENARIOS

In the system model, a large factory is considered with multiple departments such as control, production, and manufacturing. The departments have their array of equipment such as sensors, actuators, etc., that can capture and exchange information with each other for coordination of the tasks. Each department in the factory is responsible for the mutually exclusive tasks, however, for the joint tasks the departments can coordinate with each other. For example, department

x can have its process P_x that is private to department x. Similarly, the department y can have its process P_y that is private to department y. However, there can be a process $P_{x,y}$ that is a joint task between x, y. The coordination between these departments is necessary to complete the process $P_{x,y}$. The factory creates a huge amount of data due to departmental processes that need to be protected by different privacy levels. We consider the factory to have an environmental monitoring department where nodes are responsible for providing sensing information about the weather conditions, such as temperature, humidity, wind speed, and many more. Based on this information, some nodes in the network can be serviced as they would be required to turn on the ventilation systems, turn off lights, and other appropriate actions based on the environmental constraints set by IIoT users in the network. We provide two different scenarios happening in the IIoT application for this department to emphasize the security required in the flow of information and the access framework required to deal with it.

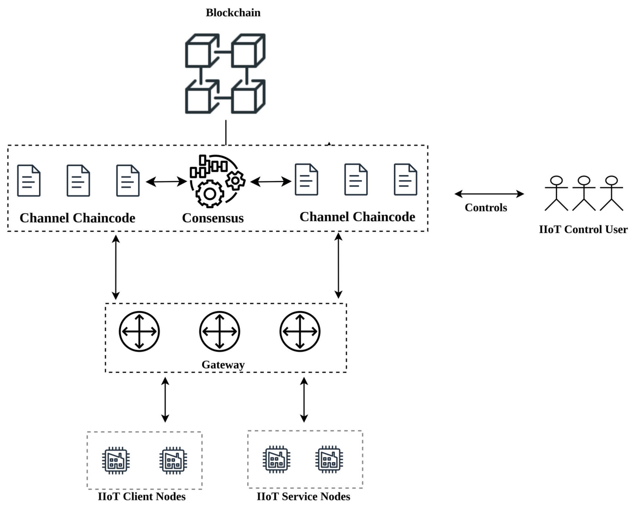


FIGURE 4. An architectural representation of the proposed framework for IIoT application.

a: SCENARIO 1

The department with environmental monitoring has nodes that serve as the client to initiate the request for the data provided by the service nodes. As the client and services are the context of the nodes, both of these contexts can be used as roles for the devices. These roles are used to determine the permissions of these nodes over the operations they will be performing in the network. This concept is extended by introducing two classes of client nodes that serve as the controller and monitor. The controller provides the appropriate actions based on the data provided by the service node. The controller would serve as the client to access the data, process it, and then translate this data into appropriate actions, and then enforce these actions by updating the policies with the help of a trusted agent in the network. For example, if the service node provides information that the temperature of the field zone rises to 50 degree, then the controller can perform actions to turn on the cooling system to lower the temperature in the field zone. Similarly, it can further enforce the actions to turn on the heating system if the temperature gets lower than the threshold. In this scenario, the controller serves as the main entity that translates the data into an actionable protocol for the agent to further provide instructions to the service nodes.

However, the monitor node is also a client node and has special use in the application. It initiates the data access and processes it only to detect if there's any hazard that exists in the field zone. The hazard is any critical situation that can damage the IIoT equipment if the situation meets the conditions. The monitor just sits idle in a normal situation and the controller is responsible for translating all the actions in this scenario.

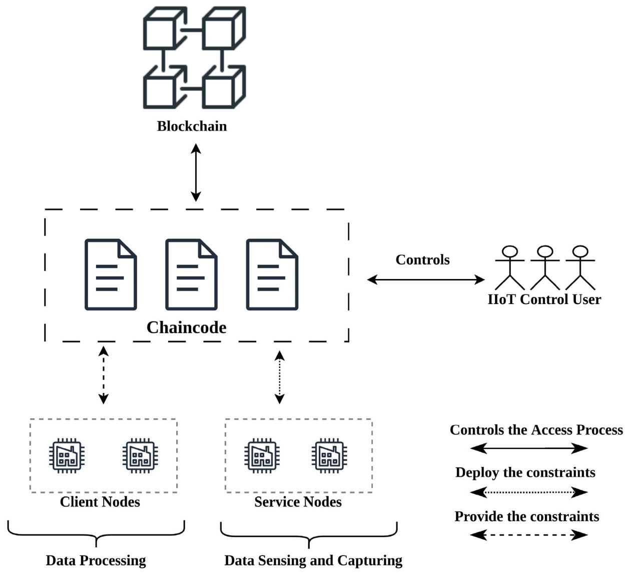


FIGURE 5. Scenario 1: Normal access process in IIoT application.

b: SCENARIO 2

In the environmental monitoring department, the service nodes produce a huge amount of data for the client nodes to process. The client nodes are either controllers or monitors in that sense that are responsible for their separate tasks. The controller translates the produced information into actions that the agent enforces via service nodes. Similarly, the monitor node can initiate the request to translate if there are critical conditions detected based on the data captured by service nodes. In that case, if the hazard is detected by the monitor, it will initiate the request to enforce the emergency response protocol to avoid any damage to valuable factory equipment. In the context of the client nodes, controller nodes have that access privilege, and monitoring requests to this privilege will certainly cause a conflict situation in the network.

To deal with this, the agent will come forward to deal with the scenario in our proposed model. As the proposed blockchain is permissioned, the nodes will only be able to operate when they get permission to do so. The conflict can be efficiently resolved by revoking the controller's permission for this scenario and will be delegated to monitor. The monitor will initiate and enforce the emergency response system to deal with the critical situation in the field zone. The network will come back to its previous state once the critical situation is tackled and there is no need to enforce the emergency response. The agent will then restore the permission to the controller node for its normal operations.

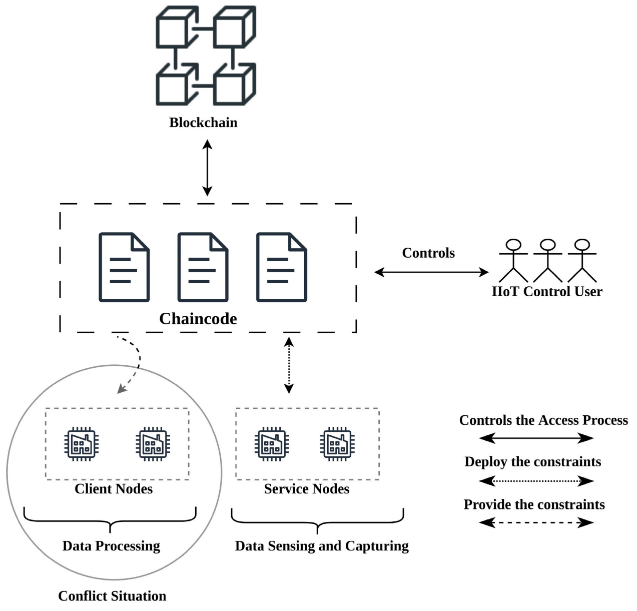


FIGURE 6. Scenario 2: Conflict in access process in IIoT application.

c: ASSUMPTIONS IN THE SCENARIOS

Since the model is based on the blockchain, the proposed system can deal with third-party solutions to further enhance the features it provides. To make the problem clear and focused, we deal with improvements in the RBAC to make it more suitable for IIoT use cases. Privacy preservation, node authentication, and other operations can be added to the proposed framework by using state-of-the-art cryptographic solutions.

The conflicting scenario that we present in the section IV-C0b is not only the case. In typical IIoT applications, mobility and dynamic nature will rapidly change the state of the network thus there can be many such situations where conflicts can arise in the policies. However, with the help of adaptive policies, risk profiles and domain segregation of policies can help to proactively detect such situations to revoke overlapping permission to deal with it.

D. CHAINCODE ARCHITECTURE

We use the chaincode functionality provided by the HF to handle the ACL process on the blockchain. To better understand the architecture, a parameter representation is provided in Table 3. However, three types of chaincode are mainly responsible for dealing with the process requirement. The three chaincodes include device contract (DC), access contract (AC), and policy contract (PC). The DC is responsible for validating the device credentials with the help of CA and providing functions for the service and client devices to read and write IIoT data in the network. The PC is assigned to perform the access delegation, and policy management in the blockchain network. Similarly, the PC is responsible for the role and permission management of each device that participates in the network. The details of each chaincode in the blockchain are as follows:

TABLE 3. Symbolic representation of the parameters used in the chaincode.

Symbol	Descriptiom
CA	Certificate Authority
$Cert$	Certificate file
CL	Client Node
PC	Policy Contract
DC	Device Contract
AC	Access Contract
ctx	Chaincode Context
tx	Transaction
SR_i	Subject Roles
SP_i	Subject Permissions
$F(x)$	Chaincode Func.
SDB	Blockchain state database
RP	RBAC Policy

1) **DC**: A device contract that is responsible for validating the device certificates and providing functions for the client or service nodes to add and read data to SDB. The contract provides $F(x)$ of *addAsset()* and *readAsset()* for the nodes. The DC has two input parameters such as device ID, and a record for which the $F(x)$ adds the data to the SDB or queries it from the SDB as per the given device ID in the parameter.
2) **PC**: The policy contract is compiled by the agent nodes and deployed to the blockchain. The PC is responsible for providing the policy management in the network. The admin defines the *RP* initially for all the nodes and sends the request to add it to the blockchain system. The data is signed with the public keys of the agent nodes and all requests are encrypted with private. This provides the mechanism for validation of the identity of the IIoT user in the network. The identity validation is done using the agents by public keys of the IIoT user and decrypting with its private key.
   However, the PC provides *addRole(), deleteRole()* for the user to update the *RP* state in the PC. For *AddRole()* the input parameter is role id, and set of *SP*.
   validatePolicy(): This method provides the validation of the *RP* in the blockchain system. For the valid *RP*, it must have proper definitions of *SR* and *SP*. The method checks the appropriate types of each definition in JSON format and returns the boolean. A policy sample is shown in Table 4 and the process is shown in Algorithm 1. This process is necessary for the cases

TABLE 4. A sample policy JSON in the policy contract.

RP.JSON
"SR": { "UserId": "11111111112", "Role": "controller", "Group": "service"},
"SP": { "DeviceId": "a100000001", "MAC": "00:11:22:33:44:55","Read": true, "Write": true, "Execute": false },
"AE": { "CreatedTime": 1645680000, "EndTime": 1645766400, "AllowedIP": ".*.1.1"}*

Algorithm 1 PolicyContract.*validatePolicy()*: Check Policy in the Blockchain

```
Data: RP
Result: Boolean
1  initialization of chaincode interface
2  SR, SP ← Parse(RP)
3  validate ← False
4  for r in SR do
5      if $r \in \{id, role, group\}$ then
6          validated ← True
7      end
8  end
9  for p in SP do
10     if $p \in \{id, MAC, permissions\}$ then
11         validated ← True
12     end
13 end
14 return validated
```

Algorithm 2 Policy Contract.*addPolicy()*: Add Policy in the Blockchain

```
Data: RP
Result: Error or true
1  initialization of chaincode interface
2  stub chaincodeStub ← Invoke()
3  if checkPolicy(RP) == False then
4      return Error("Invalid")
5  else
6      id → Sha256(RP.SP, RP.SR);
7      err → stub.PutStateChange(id, RP);
8      if err then
9          return True
10     else
11         return Error(err.Text)
12     end
13 end
```

where a new policy is being added in the chaincode using *addPolicy()* shown in Algorithm 2.

updatePolicy(): This method provides an interface for both agent and user to update the *RP* by adding new *SP* or *SR* in the current context of *RP*. The method would use the atomic methods of *addRole(), deleteRole()* and then update the *ctx* in the *SDB*. The other methods involve the *queryPolicy()* and *deletePolicy()* where both are related to the query and delete the *ctx* of *SDB* in PC. *CouchDB* is used as *SDB* for PC to facilitate rich query operators for PC to fetch the *SP* and *SR* whenever required by the agent, user, or other chaincode in the blockchain.

revokePermission(): This method provides the interface for the agent nodes only to manipulate the *SP* in the current state of *RP* and remove the recovable permissions to bypass the conflict in the *RP*.

3) **AC**: This chaincode is responsible for managing the access of client nodes to service nodes in the network. The service node compiles of deploys the contract on the blockchain when the request is initiated by the PC. The AC can further check if the role exists in the chaincode by querying the *SDB* of the blockchain using *getRole()* by client id as a parameter to the function. Similarly, when the roles are fetched, the chaincode can then again query the *SP* against the fetched role. The summary of the main functions in this chaincode is as follows:

checkAccess(): Verifies if the access is allowed against the fetched role and assigned *SP* in the *ctx*. If the access is allowed, it returns the boolean and vice versa.

addAccess(): Adds the access of the client to service nodes as requested by the agent after determining the possible conflict in the *RP*. The conflicts are determined by querying the role hierarchy from the *SDB* and then detecting the overlapping permissions using the *getOverlappingPermissions()* bypassing the role ID of the client node. The permissions are removed in case there exists any overlapping permission. That way, a new access level is added and *SP* is updated in the *ctx* of the blockchain.

delegateAccess(): Responsible for delegating the access privileges of the client node to another node assuming that both clients are the same and have the same access privileges provided to a new client in the network. This method is invoked by the PC agent on behalf of the IIoT user in cases where delegation is required for smooth operations in the network. The process is shown in Algorithm 3

E. ACCESS CONTROL PROCESS WORKFLOW

For the ACL on the network, we use the interaction of three chaincodes to manage the flow of information with protected privacy levels between the nodes in the IIoT application. To start with the process, the IIoT user must feed the *RP* to the PC to validate the *RP* if it meets the satisfactory criteria. Initially, the whole blockchain network would be set up, CA will generate the certificates for the nodes in the

Algorithm 3 AccessContract.*delegateAccess()*: Delegate the Access in the Blockchain

```
Data: $ctx, RP, deviceid, role$
Result: $Error$ or $True$
1  initialization of chaincode interface
2  $SR, SP \leftarrow Parse(RP)$
3  if $role \notin SR$ then
4      return $err.Text$
5  end
6  $rolesJSON \leftarrow ctx.stub.getState(role)$
7  if $\neg roleJSON$ then
8      return $err.Text$
9  end
10 $\hat{SP} \leftarrow SP + roleJSON.permission$
11 $\hat{RP} \leftarrow RP + SP$
12 $err\ leftarrow\ ctx.stub.putState(RP)$
13 if $\neg err$ then
14     return $err.Text$
15 end
16 return $True$
```

networks and each node will then deploy their chaincodes in the blockchain. For two scenarios represented in Figure 5 and 6, the ACL process for scenario 1 would be as follows:

1) **Process 1**: The IIoT user specifies the set of *SR* and *SP* to define the *RP* and feed it to PC and *SDB*. A basic representation of the *RP* is represented in Equation 4.
$$RP \rightarrow \{SR, SP, \} \tag{4}$$
The *RP* is then validated by the PC for possible missing constraints and committed to *SDB* if the PC validates the policy successfully. This process can be represented as shown in Equation 5.
$$PC(RP) \rightarrow \{Ledger, SDB\} \tag{5}$$
2) **Process 2**: Client node requests to process the chunk of data by requesting DC and this request is processed against the access, and then access is granted if the *SR* has permission for it. This is done by invoking the AC to get the *RP* and access is provided to the client for data read.
3) **Process 3**: The AC gets invoked by the DC with the role ID of the client, for which it checks the access privileges of the client in the *RP*. The AC further invoked the PC to get the *RP* and permissions are checked.
4) **Process 4**: The permissions are checked, and the client request is served with the requested operation by querying the data from the *SDB*, and transaction is posted in the blockchain ledger. This process will call back to AC to grant access and AC will reflect DC for the deployment of the transaction.

The processes for the service nodes would remain the same for the service nodes but a slight difference in the operation requested by the service node in the blockchain. The service nodes would operate to add data in the blockchain invoking the DC with the add operation and workflow would be the same as demonstrated above. The chaincodes interact with each other for the access control process in this scenario and a similar interaction between the chaincode would occur in scenario 2 with a slight difference in Process 4. The activities done in these steps are shown in Figure 7

For Scenario 2 as represented in Figure 6, process 4 will involve the agent with possible conflicts in the *RP* got by the PC. In that case, the agent will find the overlapping *SP* and will revoke to bypass the conflicting situation. After revoking the permissions, the callbacks will initiate from AC to DC, and the process will deploy the transaction on the ledger updating the *ctx* of the blockchain. Similarly, the updated *RP* will be committed to *SDB*. However, for this scenario, the client policy will be restored to its previous state after the transaction is posted successfully by the DC of the client node. The process is shown in Algorithm 4 and workflow of this operation is shown in Figure 8.

Algorithm 4 PolicyContract.*manageConflict()*: Manage the Policy Conflict in the Blockchain

```
Data: $ctx, RP$
Result: $Error$ or $True$
1  initialization of chaincode interface
2  $SR, SP \leftarrow Parse(RP)$
3  for $r$ in $SR$ do
4      $cr \rightarrow checkOverlappingPermission(ctx, r)$
5      if $cr$ then
6          $err \leftarrow revokePermission(cr, r)$
7          if $\neg err$ then
8              return $err.Text$
9          end
10     end
11 end
12 $state\ leftarrow\ changeState(id, ctx, role)$
13 $err\ leftarrow\ ctx.stub.putState(id)$
14 if $\neg err$ then
15     return $err.Text$
16 end
17 return $True$
```

For the atomic operations, where there is any change required by the user to update the *RP* it can use the agent to update the *RP* using the PC and new $\hat{RP}$ will be committed to the ledger and *SDB*. This process would remain the same for the operation of delegate access where the authority of one device can be delegated to another device. In that case, the IIoT user can invoke the AC using the agent interface and AC would be invoked. The policy will be updated by PC just same as previous. All operations in the blockchain for ACL will be done with the interaction of these chaincodes where client or service nodes can only interact with DC and DC will invoke

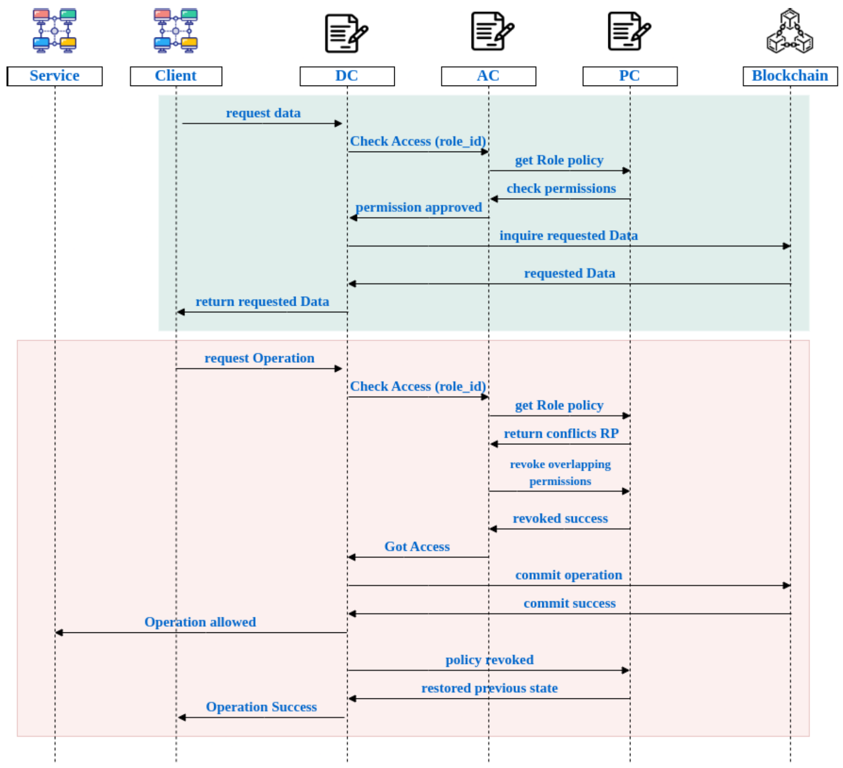


FIGURE 7. Activity diagram of access control framework in the blockchain.

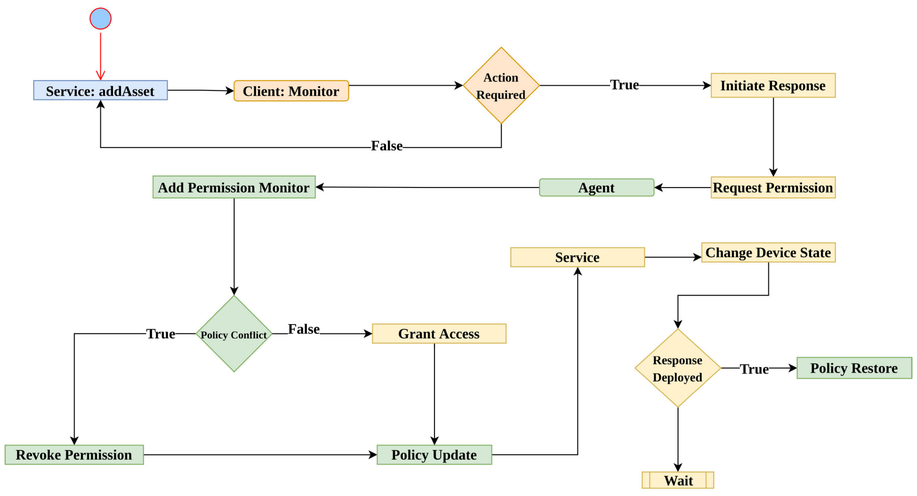


FIGURE 8. Access control workflow for conflict management in the network.

the other chaincodes thus ensuring the isolation and privacy for each chaincode involved in overall ACL process.

V. IMPLEMENTATION

This section provides details about the implementation of the blockchain network and how ACL is added on top of HF to provide a simulation of the proposed framework. The framework will be then evaluated for qualitative and quantitative comparisons of computational time between the different SOTA studies [30], [32]. To implement the HF, the dependencies and environment are provided in Table 5, and development environment dependencies are provided in Table 6. The implementation is divided into two parts: i) Environment setup and ii) Access control implementation. The environment section provides the structure of our proposed methodology and chaincode installation. The second section discusses the chaincode invocation for the implementation of ACL on the top of the HF.

A. STRUCTURE IMPLEMENTATION

This section introduces the structure of the blockchain network and the proposed framework. In the network, there

TABLE 5. Hardware and software dependencies.

Hardware	Description
OS	Ubuntu v23.10
CPU	Intel Core i7-12800H Turbo 4.0GHz P-Core
Memory	64GB DDR5, 4800MHz
Hard Disk	256GB SSD
Software	
hyperledger fabric	v2.5.4
Sinon-chai	v3.2.0
node js	v21.6.1
docker	v25.0.3
docker-compose	v2.24.3

TABLE 6. Structure of the proposed framework.

Images	Description	Number
fabric/CA	CA Node	2
fabric/SDB	Couch State Database	4
fabric/peer	Peer Node	4
fabric/orderer	Orderer Node	1
fabric/fabric-tools	Fabric Tools	1
fabric/chaincode/DC	Device Contract Chaincode	4
fabric/chaincode/PC	Policy Contract Chaincode	4
fabric/chaincode/AC	Access Contract Chaincode	4

are a total of eight different types of network images running on the docker as shown in Table 6. The environment build is based on the many steps to establish the interaction between these images running in isolation. To build the network, the HF cryptographic tool is used initially to generate the root certificates and secret key pairs for nodes in the network. The secret keys provide membership service to peer nodes running in the network by assigning it identity and permission to join the network.

The second step involves mounting these generated certificates and secret key pairs to docker images of CA. The effect of these certificates will take place when the container running on the docker. After this, the other nodes in the network can validate their identity using their signature to the CA.

The third step involves deploying the identity of all nodes in the blockchain network. For this, the configtxgen tool is used to generate the genesis block that would contain the configurations of the nodes and channels in the package form. This genesis block is written into the blockchain network and identity information moves to the network by this. Once the genesis block is written into the blockchain, the information becomes immutable and tamper-proof. After this, the dependencies of the docker-compose are installed in a specified order. Once all the container runs successfully, the peer nodes will be added to the network.

The next step is to move all the chaincodes to the blockchain using the HF client by running a command to package the chaincode to a peer node in the network. All three chaincodes are mounted on peer nodes in their channel and peer nodes are upgraded simultaneously one by one.

B. RBAC IMPLEMENTATION

To implement the *RP* the chaincodes are installed on peer nodes. Peer nodes in the HF are responsible for achieving the consensus in the network. The Node JS client is used to connect the peer node to query the *SDB* by achieving a consensus using RAFT [32] that achieves better results than the PBFT in real-time scenarios [32]. The steps of chaincode are followed to add the *SP* by invoking the PC chaincode method *addPolicy()* as shown in Algorithm 2 and verify the addition by invoking the *queryPolicy()*. The result of *addPolicy()* invocation is shown in Figure 9 and Figure 10 shows the results of method *queryPolicy()* invocation.

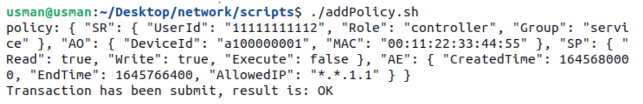

```
usman@usman:~/Desktop/network/scripts$ ./addPolicy.sh
policy: { "SR": { "UserId": "11111111112", "Role": "controller", "Group": "servi
ce" }, "AO": { "DeviceId": "a100000001", "MAC": "00:11:22:33:44:55" }, "SP": { "
Read": true, "Write": true, "Execute": false }, "AE": { "CreatedTime": 164568000
0, "EndTime": 1645766400, "AllowedIP": "*.*.1.1" } }
Transaction has been submit, result is: OK
```

FIGURE 9. Deployment results of add policy in the blockchain.

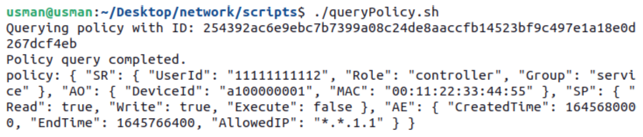

```
usman@usman:~/Desktop/network/scripts$ ./queryPolicy.sh
Querying policy with ID: 254392ac6e9ebc7b7399a08c24de8aaccfb14523bf9c497e1a18e0d
267dcf4eb
Policy query completed.
policy: { "SR": { "UserId": "11111111112", "Role": "controller", "Group": "servi
ce" }, "AO": { "DeviceId": "a100000001", "MAC": "00:11:22:33:44:55" }, "SP": { "
Read": true, "Write": true, "Execute": false }, "AE": { "CreatedTime": 164568000
0, "EndTime": 1645766400, "AllowedIP": "*.*.1.1" } }
```

FIGURE 10. Deployment results of query policy in the blockchain.

In case, where network dynamics get changed, the policy will require the updation to adjust itself with the changes. In that case, the updation of policy is accommodated by the *updatePolicy()* , and invocation results are shown in Figure 11.

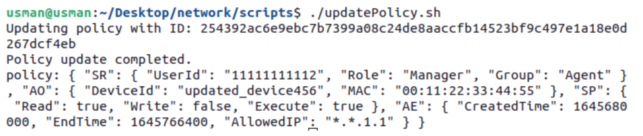

```
usman@usman:~/Desktop/network/scripts$ ./updatePolicy.sh
Updating policy with ID: 254392ac6e9ebc7b7399a08c24de8aaccfb14523bf9c497e1a18e0d
267dcf4eb
Policy update completed.
policy: { "SR": { "UserId": "11111111112", "Role": "Manager", "Group": "Agent" }
, "AO": { "DeviceId": "updated_device456", "MAC": "00:11:22:33:44:55" }, "SP": {
 "Read": true, "Write": false, "Execute": true }, "AE": { "CreatedTime": 1645680
000, "EndTime": 1645766400, "AllowedIP": "*.*.1.1" } }
```

FIGURE 11. Deployment results of update policy in the blockchain.

Similarly, the service nodes can add data in the network using the device contract by invoking the *addAsset()*, and results are shown in Figure 12. The client nodes in the network can then use this data by invoking the *readAsset()*. The *readAsset()* operation is performed after the AC checks the access privileges and grants it successfully.

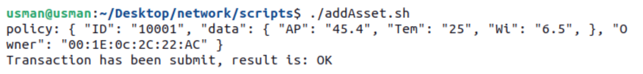

```
usman@usman:~/Desktop/network/scripts$ ./addAsset.sh
policy: { "ID": "10001", "data": { "AP": "45.4", "Tem": "25", "Wi": "6.5", }, "O
wner": "00:1E:0c:2C:22:AC" }
Transaction has been submit, result is: OK
```

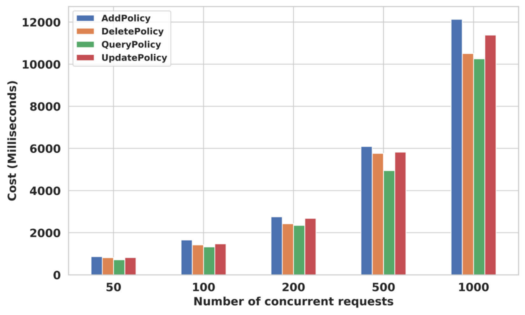
FIGURE 12. Deployement result of add asset in the blockchain.

VI. RESULTS AND DISCUSSION

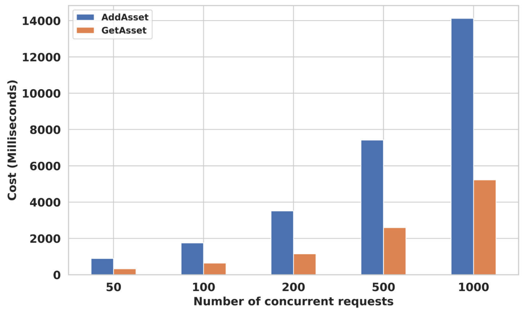
For the evaluation of the proposed model, the comprehensive experiments are designed to test the system processing time for various functions related to access control. The results are discussed in two sections, where in the first section, the simulation results of the proposed system for different operations performed to test the cost consumed by the network are discussed. In the later section, a comparison is presented between the proposed model and baseline studies to emphasize the need for an extensive RBAC model and its effectiveness in reducing the computational overhead. The source code of the proposed framework is open source and available on GitHub at: https://bit.ly/3x7luUV

A. SIMULATION RESULTS

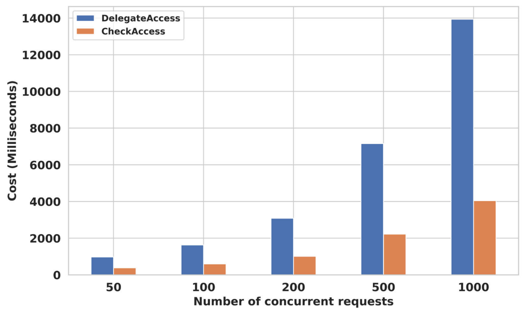
The cost is defined as the processing time consumed by the network to respond to the operations of virtual clients. The virtual clients are set by concurrent requests by multithreaded invocation of the chaincode for different operations. The number of virtual clients is set to 50, 100, 200, 500, and 1000 to test the cost of the network under normal to heavy operational load.



(a) Policy Contract



(b) Device Contract



(c) Access Contract

FIGURE 13. Total Cost of the proposed system for virtual concurrent requests.

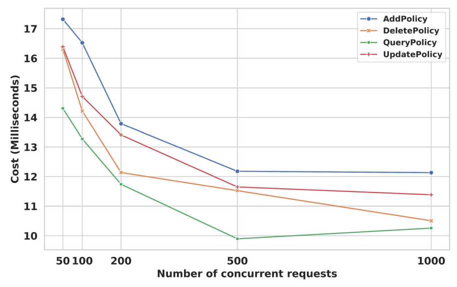
The increasing number of virtual clients would help to test the scalability of the system which is crucial in IIoT applications. For the simulation results, the results are divided into two separate experiments where in the first case, the system cost for each contract is considered for their operations for the specified number of virtual clients. In the second case, the system is tested for the average cost for each contract for its operations. The results are represented in figures 13 and 14, where figure 13 represents the individual cost and 13 represents the average cost of the proposed system. For the PC, statistical results for both individual and average cases are shown in figures 13a and 14a. The cost of the proposed system is observed to increase with increasing virtual clients, however, the trend of this growth is linear, and with a certain amount of the clients in the network, the cost tends to be stable. The average time per transaction is observed to decrease with an increased number of virtual clients as shown in Figure 14a.

The reason behind this phenomenon is likely to be the caching for the recent results to be fetched by the chaincode in the network from the state database. Similarly, the cost of the blockchain network would increase with an increased number of clients if the operations requests by clients are different. Similarly, the network cost would also depend on the complexity of the tasks involved in the chaincode. It is observed from Figure 13a and 14a that the write (add, update) operations are more complex than the read (get, read) operations as the cost for write operations is higher than read operations. This is because external data needs to be validated by the orderer nodes before committing it to the blockchain ledger. However, for individual responses, the cost increases concerning the number of clients in the network due to the validation required by each node in the network. The same trend is observed for DC operations where service and client nodes request and serve the data to and from the network. Figures 13b and 14b show the average time consumed per transaction in the blockchain for both individual and average cases. For DC operations, the read operation tends to show the same pattern as it was in PC.

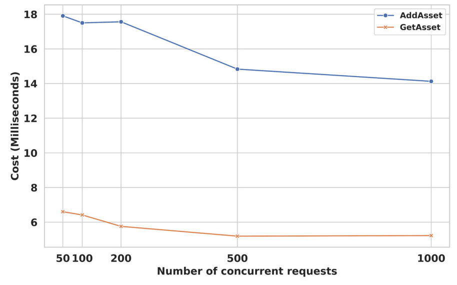
For the AC, the system is tested for ACL to evaluate the system for the cost it takes to check access against the request by client nodes in the network. The concurrent requests were tested for delegate access, get role, and conflict management, and average cost results are shown in Figure 13c. From Figure 14c, it can again observed the same phenomenon where read operations incur less cost than write operations and transaction time is increased with an increased number of virtual clients. However, the growth behavior is independent of the size of the virtual client thus the average time shows the downtrend in Figure 14c.

B. BASELINE COMPARISONS

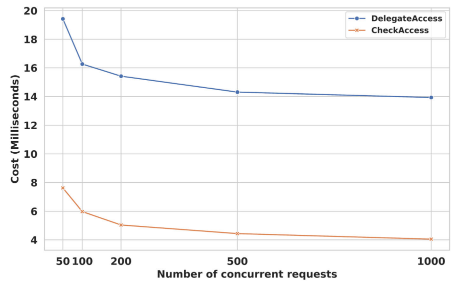
RBAC is traditionally a simple ACL mechanism in comparison to ABAC systems, thus it fails to meet the requirements of the IIoT network. The proposed framework improves the RBAC lacking to make it suitable for IIoT applications. Theoretically, it is believed that using RBAC can overcome the high computation overhead of the ABAC system. To test



(a) Policy Contract



(b) Device Contract



(c) Access Contract

FIGURE 14. **Average cost of the proposed system for virtual concurrent requests.**

this hypothesis, the computation cost of the proposed framework is compared with the study proposed in [30] and [32] that used the ABAC for information flow control in blockchain networks. For the comparison, the concurrent requests for virtual clients are considered as 50, 100, and 200 as common ground between these studies. These concurrent virtual clients are then used to compare the studies for write operations for all chaincodes to test the studies for their average computation overhead.

TABLE 7. **Comparison of average computation time (seconds) for access control operations.**

# of reqs.	Study	Add Policy	Update Policy	Add Asset	Get Access
50	[32]	0.125	0.120	0.125	0.075
	[30]	0.120	0.115	0.100	0.120
	Proposed	**0.017**	**0.017**	**0.018**	**0.008**
100	[32]	0.130	0.128	0.130	0.080
	[30]	0.108	0.115	0.120	0.080
	Proposed	**0.016**	**0.015**	**0.018**	**0.006**
200	[32]	0.140	0.135	0.110	0.078
	[30]	0.145	0.118	0.155	0.075
	Proposed	**0.014**	**0.013**	**0.018**	**0.006**

In this test, the proposed system outperforms the baseline studies significantly for write operations of chaincodes. Table 7 shows the performance results of studies for their chaincode computation overhead. While both ABAC studies represented consistent performance for virtual clients, the proposed RBAC system time overhead is significantly less than ABAC. This significant drop in time overhead demonstrates the efficiency of RBAC to provide consistent control flow in IIoT applications if RBAC is tailored to meet the requirements of IIoT use cases.

VII. CONCLUSION

This paper proposed an access control framework using a layered architecture of smart contracts to control the information flow in the IIoT use cases. The proposed system takes advantage of blockchain technology such as tamper-proof transactions, proof of identity, and scalability that better deal with the requirements of IIoT. With HF, the proof of identity is achieved using the CA that can add or cancel the device credentials. Similarly, using the cryptographic hashing in the orderer node adds the hash of each block in chains to make it impossible to tamper and alter. These inherited benefits of the blockchain help the system to achieve the more robust functionality that traditional centralized system fails to do so. However, the ABAC system implemented on the top of the blockchain using smart contracts suffers from high computational overhead. This study emphasizes the benefits of a simple RBAC model to make it fit for the IIoT use case where it can deal with dynamic situations in the network. In this paper, a use case scenario is proposed where the IIoT network is segregated into domains that interact with each other for information exchange. To control the information flow, the smart contracts are used to deal with access evaluation for each participating node in the network. To implement the proposed access control, the hyper ledger fabric and chaincodes are used in the layered architecture. A detailed presentation of the network system model, blockchain establishment, and a statistical simulation evaluation of the proposed model are presented for various numbers of participating nodes in the network. Similarly, a detailed comparison of the proposed model is also provided to demonstrate the efficiency of the RBAC over ABAC for computational overhead. The proposed RBAC deals with the dynamic roles and scalability in the network. However, the RBAC systems are limited to dealing with role explosion and manual role assignment to participating nodes in the network. Similarly, when the network grows to a certain point, the RBAC systems become more vulnerable to policy conflicts. For future work, we would consider implementing the RBAC systems to deal with these issues in the context of IIoT use cases.

REFERENCES

[1] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Internet of Things (IoT) security intelligence: A comprehensive overview, machine learning solutions and research directions," *Mobile Netw. Appl.*, vol. 28, no. 1, pp. 296–312, Feb. 2023.

[2] X. Zhu, J. Huang, and C. Qi, "Modeling and analysis of malware propagation for IoT heterogeneous devices," *IEEE Syst. J.*, vol. 17, no. 3, pp. 3846–3857, Sep. 2023.

[3] C. Maraveas, D. Piromalis, K. G. Arvanitis, T. Bartzanas, and D. Loukatos, "Applications of IoT for optimized greenhouse environment and resources management," *Comput. Electron. Agricult.*, vol. 198, Jul. 2022, Art. no. 106993.

[4] A. Vaclavova, P. Strelec, T. Horak, M. Kebisek, P. Tanuska, and L. Huraj, "Proposal for an IIoT device solution according to Industry 4.0 concept," *Sensors*, vol. 22, no. 1, p. 325, Jan. 2022.

[5] S. Javed, M. Usman, F. Sandin, M. Liwicki, and H. Mokayed, "Deep ontology alignment using a natural language processing approach for automatic M2M translation in IIoT," *Sensors*, vol. 23, no. 20, p. 8427, Oct. 2023.

[6] J. Leng, W. Sha, B. Wang, P. Zheng, C. Zhuang, Q. Liu, T. Wuest, D. Mourtzis, and L. Wang, "Industry 5.0: Prospect and retrospect," *J. Manuf. Syst.*, vol. 65, pp. 279–295, Oct. 2022.

[7] S. Javed, S. Javed, J. v. Deventer, H. Mokayed, and J. Delsing, "A smart manufacturing ecosystem for industry 5.0 using cloud-based collaborative learning at the edge," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, May 2023, pp. 1–6.

[8] C. Bayılmış, M. A. Ebleme, Ü. Çavuşoğlu, K. Küçük, and A. Sevin, "A survey on communication protocols and performance evaluations for Internet of Things," *Digit. Commun. Netw.*, vol. 8, no. 6, pp. 1094–1104, Dec. 2022.

[9] M. Li, P. Pei, F. R. Yu, P. Si, Y. Li, E. Sun, and Y. Zhang, "Cloud-edge collaborative resource allocation for blockchain-enabled Internet of Things: A collective reinforcement learning approach," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 23115–23129, Nov. 2022.

[10] C. Shao, Y. Yang, S. Juneja, and T. GSeetharam, "IoT data visualization for business intelligence in corporate finance," *Inf. Process. Manage.*, vol. 59, no. 1, Jan. 2022, Art. no. 102736.

[11] R. Mehannaoui, K. N. Mouss, and K. Aksa, "IoT-based food traceability system: Architecture, technologies, applications, and future trends," *Food Control*, vol. 145, Mar. 2023, Art. no. 109409.

[12] M. Nasir, K. Muhammad, A. Ullah, J. Ahmad, S. W. Baik, and M. Sajjad, "Enabling automation and edge intelligence over resource constraint IoT devices for smart home," *Neurocomputing*, vol. 491, pp. 494–506, Jun. 2022.

[13] M. Usman, M. S. Sarfraz, U. Habib, M. U. Aftab, and S. Javed, "Automatic hybrid access control in SCADA-enabled IIoT networks using machine learning," *Sensors*, vol. 23, no. 8, p. 3931, Apr. 2023.

[14] A. Irshad, G. A. Mallah, M. Bilal, S. A. Chaudhry, M. Shafiq, and H. Song, "SUSIC: A secure user access control mechanism for SDN-enabled IIoT and cyber physical systems," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 16504–16515, Sep. 2023.

[15] S. S. Ullah, V. Oleshchuk, and H. S. G. Pussewalage, "A survey on blockchain envisioned attribute based access control for Internet of Things: Overview, comparative analysis, and open research challenges," *Comput. Netw.*, vol. 235, Nov. 2023, Art. no. 109994.

[16] M. Shakarami and R. Sandhu, "Role-based administration of role-based smart home IoT," in *Proc. ACM Workshop Secure Trustworthy Cyber-Physical Syst.*, Apr. 2021, pp. 49–58.

[17] R. Saha, G. Kumar, M. Conti, T. Devgun, T.-H. Kim, M. Alazab, and R. Thomas, "DHACS: Smart contract-based decentralized hybrid access control for industrial Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3452–3461, May 2022.

[18] Rishikesh and D. Sinha, "Traditional and blockchain based IoT and IIoT security in the context of agriculture: A survey," *Wireless Pers. Commun.*, vol. 133, no. 4, pp. 2267–2295, Dec. 2023.

[19] S. Mathur, A. Kalla, G. Gür, M. K. Bohra, and M. Liyanage, "A survey on role of blockchain for IoT: Applications and technical aspects," *Comput. Netw.*, vol. 227, May 2023, Art. no. 109726.

[20] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain-based distributed application for multimedia system using hyperledger fabric," *Multimedia Tools Appl.*, vol. 83, no. 1, pp. 2473–2499, Jan. 2024.

[21] K. Y. Najmi, M. A. AlZain, M. Masud, N. Z. Jhanjhi, J. Al-Amri, and M. Baz, "A survey on security threats and countermeasures in IoT to achieve users confidentiality and reliability," *Mater. Today, Proc.*, vol. 81, pp. 377–382, Jul. 2023.

[22] U. R. Saxena and T. Alam, "Provisioning trust-oriented role-based access control for maintaining data integrity in cloud," *Int. J. Syst. Assurance Eng. Manage.*, vol. 14, no. 6, pp. 2559–2578, Dec. 2023.

[23] M. T. de Oliveira, Y. Verginadis, L. H. A. Reis, E. Psarra, I. Patiniotakis, and S. D. Olabarriaga, "AC-ABAC: Attribute-based access control for electronic medical records during acute care," *Expert Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119271.

[24] D.-Y. Kim, S. Lee, M. Kim, and S. Kim, "Edge cloud selection in mobile edge computing (MEC)-aided applications for industrial Internet of Things (IIoT) services," *Comput. Syst. Sci. Eng.*, vol. 47, no. 2, pp. 2049–2060, 2023.

[25] T. Wu, G. Jourjon, K. Thilakarathna, and P. L. Yeoh, "MapChain-D: A distributed blockchain for IIoT data storage and communications," *IEEE Trans. Ind. Informat.*, 2023.

[26] S. Zeba, P. Suman, and K. Tyagi, "Types of blockchain," in *Distributed Computing to Blockchain*. Amsterdam, The Netherlands: Elsevier, 2023, pp. 55–68.

[27] T. Guggenberger, J. Sedlmeir, G. Fridgen, and A. Luckow, "An in-depth investigation of the performance characteristics of hyperledger fabric," *Comput. Ind. Eng.*, vol. 173, Nov. 2022, Art. no. 108716.

[28] G. Yang, K. Lee, K. Lee, Y. Yoo, H. Lee, and C. Yoo, "Resource analysis of blockchain consensus algorithms in hyperledger fabric," *IEEE Access*, vol. 10, pp. 74902–74920, 2022.

[29] Y.-F. Wen and C.-M. Hsu, "A performance evaluation of modular functions and state databases for hyperledger fabric blockchain systems," *J. Supercomput.*, vol. 79, no. 3, pp. 2654–2690, Feb. 2023.

[30] H. Liu, D. Han, and D. Li, "Fabric-IoT: A blockchain-based access control system in IoT," *IEEE Access*, vol. 8, pp. 18207–18218, 2020.

[31] S. Liu, L. Chen, H. Yu, S. Gao, and H. Fang, "BP-AKAA: Blockchain-enforced privacy-preserving authentication and key agreement and access control for IIoT," *J. Inf. Secur. Appl.*, vol. 73, Mar. 2023, Art. no. 103443.

[32] Y. Feng, W. Zhang, X. Luo, and B. Zhang, "A consortium blockchain-based access control framework with dynamic orderer node selection for 5G-enabled industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2840–2848, Apr. 2022.

[33] D.-H. Shih, T.-W. Wu, M.-H. Shih, G.-W. Chen, and D. C. Yen, "Hyperledger fabric access control for industrial Internet of Things," *Appl. Sci.*, vol. 12, no. 6, p. 3125, Mar. 2022.

[34] S. Sutradhar, S. Karforma, R. Bose, S. Roy, S. Djebali, and D. Bhattacharyya, "Enhancing identity and access management using hyperledger fabric and OAuth 2.0: A block-chain-based approach for security and scalability for healthcare industry," *Internet Things Cyber-Phys. Syst.*, vol. 4, pp. 49–67, Jul. 2024.

[35] X. Hao, W. Ren, Y. Fei, T. Zhu, and K. R. Choo, "A blockchain-based cross-domain and autonomous access control scheme for Internet of Things," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 773–786, Mar. 2023.

[36] P. Kamboj, S. Khare, and S. Pal, "User authentication using blockchain based smart contract in role-based access control," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 2961–2976, Sep. 2021.

[37] T. Zaidi, M. Usman, M. U. Aftab, H. Aljuaid, and Y. Y. Ghadi, "Fabrication of flexible role-based access control based on blockchain for Internet of Things use cases," *IEEE Access*, vol. 11, pp. 106315–106333, 2023.

[38] S. Figueroa-Lorenzo, J. Añorga, and S. Arrizabalaga, "Methodological performance analysis applied to a novel IIoT access control system based on permissioned blockchain," *Inf. Process. Manage.*, vol. 58, no. 4, Jul. 2021, Art. no. 102558.

[39] S. Luo, T. Hu, N. Han, and Y. Qian, "A smart-contract-based policy-domain access control framework for distributed industrial IoT," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 11427–11443, Apr. 2024.

MUHAMMAD USMAN is currently pursuing the Ph.D. degree with the FAST National University of Computer and Emerging Sciences, Chiniot-Faisalabad Campus, Pakistan. He is also a Lecturer with the Department of Computer Science, FAST National University of Computer and Emerging Sciences. His research interests include blockchain, cyber security, predictive analytics, and machine learning applications for cross-domain applications.

MUHAMMAD SHAHZAD SARFRAZ (Senior Member, IEEE) received the Ph.D. degree from the Asian Institute of Technology (AIT), Thailand. He was a Visiting Researcher with the Digital Image Processing Laboratory, National Institute of Informatics (NII), Tokyo, Japan. He is currently a Professor in computer science with the FAST National University of Computer and Emerging Sciences. He is also the Director of Campus with the FAST National University of Computer and Emerging Sciences, Chiniot-Faisalabad Campus. He has an extensive background and interest in using geospatial technologies to strengthen and integrate health systems. He contributed to many seminars/workshops, organized at national and international levels, which supported collaborative research in ten countries. He is a member of numerous international societies, like Telecoms Sans Frontiers for Emergency Response in Asia–Pacific, ISPRS Health, the Geological Society of America, the International Medical Geological Association, the eHealth Association of Pakistan, and American Telemedicine Association.

MUHAMMAD UMAR AFTAB received the B.S. degree from Government College University Faisalabad, in 2011, the master's degree in computer science from the National Textile University, Pakistan, in 2014, and the Ph.D. degree from the School of Information and Software Engineering, University of Electronic Science and Technology of China (UESTC), in 2020. He is currently an Assistant Professor with the Department of Computer Science, FAST National University of Computer and Emerging Sciences. His research interests include information/network security, authorization/access control (RBAC and ABAC), cryptography, and network technologies. He is serving as a reviewer and a guest editor for various renowned journals.

USMAN HABIB received the master's degree from Norwegian University of Science and Technology (NTNU), Norway, and the Ph.D. degree from the ICT Department, Technical University of Vienna, Austria. He is currently an Associate Professor and the Head of the Software Department, FAST School of Computing, National University of Computer and Emerging Sciences (NUCES), Islamabad, Pakistan. Before joining FAST-NUCES, he was with the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI), Swabi and COMSATS University, Abbottabad Campus. With over 18 years of teaching and research experience, since 2006, he has been completing various industrial projects along with serving in academia. He also actively engages in research and has authored numerous conference and journal publications. His current research interests include machine learning, data analytics, pattern recognition, security, and medical image processing.

SALEHA JAVED was born in Lahore, Pakistan, in 1990. She received the B.S.C.S. degree (Hons.), in 2012. She is currently pursuing the Ph.D. degree with Luleå Technical University, Sweden. During the master's (M.S.C..S.) education, she was selected for a DAAD funded yearlong research project with the Technical University of Kaiserslautern (TUK), Germany, for which she traveled to Kaiserslautern and worked on the Optimization of UAVs using bio-inspired approaches. Then, she is currently a Senior Research Engineer with Germany-based company SOCO Engineers, where she was also getting the Project Manager training. Along with her industry job, she continued the research work and then moved to the educational sector to focus on research even more. From 2016 to 2019, she taught as a Senior Lecturer with the University of Lahore and the University of South Asia, Pakistan. She has been working on a hybrid research domain of CPS and ML utilization for industry 4.0 and 5.0 IIoT solutions. Thus far, she has 12 articles as an outcome of her research and she continues to achieve more outcomes. Her research interests include the conceptualization and modeling of learning-based collaborative digitalization designs for smart industry use cases.