Received February 28, 2020, accepted March 12, 2020, date of publication March 23, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982542

xDBAuth: Blockchain Based Cross Domain Authentication and Authorization Framework for Internet of Things

GAUHAR ALI[1], NAVEED AHMAD[1], YUE CAO[2], (Member, IEEE), SHAHZAD KHAN[1], HAITHAM CRUICKSHANK[3], (Member, IEEE), EJAZ ALI QAZI[1], AND AZAZ ALI[4]
[1]Department of Computer Science, University of Peshawar, Peshawar 25120, Pakistan
[2]School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[3]Institute of Communication Systems, University of Surrey, Guildford GU2 7JP, U.K.
[4]School of Computer Science, China West Normal University, Nanchong 637000, China

Corresponding author: Yue Cao (871441562@qq.com)

ABSTRACT The innovation of ubiquitous and pervasive computing helps service-oriented organizations in the realization of a virtual coalition. The virtual coalition is a set of IoT domains i.e., smart homes and smart hospitals that are linked together through communication lines to share resources. Such virtual coalitions need secure cross-domain permission delegation and access control mechanisms. In existing approaches, permission delegation and access control are performed at the resource owner domain or by a single trusted third party. This single trusted third party may fail to work or compromise. Therefore, it will collapse either the whole system or the security of the system. We propose xDBAuth, a decentralized Blockchain (BC) based permission delegation and access control framework for the Internet of Things (IoT). Also, we proposed a hierarchy of local and global smart contracts that perform permission delegation and access control for both internal and external user/IoT devices. Additionally, the proposed framework preserves an external user's privacy by allowing them to get authentication in their parent IoT domains. During authentication, Proof-of-Authenticity/Integrity (PoAI) mechanism is used to find and retrieve user/IoT device platform hashes stored on local BC. After successful authentication, BC authorizes the user/IoT device based on the validation of delegation policies stored on BC. We implemented the proposed framework using Node.js. The results show that the proposed xDBAuth is a lightweight framework with less computational overhead. xDBAuth produces high throughput in an environment having a large number of concurrent requests.

INDEX TERMS Access control, blockchain, Internet of Things, permission delegation.

I. INTRODUCTION

Internet of Things (IoT) [1], consists of ubiquitous objects i.e., smart devices, sensors, etc that can gather and transfer data to each other in an automated way. The ubiquitous computing enabled devices have made IoT one of the most fundamental architecture. It has been successfully applied in various applications such as smart healthcare, Intelligent Transportation System (ITS), smart home, etc. According to the Gartner report, by the end of the year 2020, approximately 25 billion devices will be connected in IoT [2]. However, these IoT devices work under different administrations. For example, IoT devices in a smart home or smart devices in a hospital have their own administration. Such a group of user/IoT devices is called the IoT domain. These IoT domains i.e., smart homes and maintenance companies or smart hospitals can be connected to form a virtual coalition. Hence, the virtual coalition is a network of organizations connected by communications technologies to share resources.

National Health Information Network (NHIN) [3] is a virtual coalition of multiple hospitals. Such a virtual coalition enables the hospitals to open their data and services for cross-domain access. For example, a physician in hospital **A** wants to discuss a patient disease with a physician in hospital **B**. Therefore, the physician in hospital **B** needs permissions on all the devices and sensors attached to the patient body. Similarly, a virtual coalition of smart homes and maintenance companies. These virtual coalitions save time or/and money of both the parties. However, these virtual coalitions raise new challenges.

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

In the real world, virtual coalitions hire trusted third party companies for security checking. Such companies validate Service Requester (SR) credentials based on the policy defined by the Service Provider (SP). A single trusted entity is proposed in both [4] and [5] for cross-domain permission delegation and access control. Here, if the single trusted entity fails, then cross-domain access will not occur. Similarly, In conventional Public Key Infrastructure (PKI), a centralized trusted Certification Authority (CA) is used to certify the ownership of the public keys using digital certificates. The PKI security will collapse when the single trusted CA is compromised [6].

Moreover, these single trusted third parties are vulnerable to different attacks [7]. As a result, an adversary can hijack user accounts, data traffic. Such a single entity has several other threats like privacy issues, lack of transparency, insecure authentication. Furthermore, the secret key among DS and the organizations in a coalition is a single point of failure [8]. Therefore, a decentralized, secure, and lightweight mechanism is required for cross-domain permission delegation and access control. Also, the proposed mechanism needs to develop trust among coalition partners, meanwhile without a single trusted entity.

We propose xDBAuth, a decentralized blockchain (BC) based framework for cross-domain permission delegation and access control. In a virtual coalition of the IoT domains, each IoT domain has a local BC and single local smart contract to manage the internal resources i.e., user/IoT devices. The global BC is an overlay network formed by IoT domains in the coalition. It has a single global smart contract. Global smart contract performs cross-domain permission delegation and access control of external user/IoT devices. The proposed framework provides strong enough privacy protection because SR gets authentication from his parent's domain. Similarly, it provides secure user/IoT device authentication based on the verification of platform hashes generated TPM (Trusted Platform Module). A Proof-of-Authenticity/Integrity (PoAI) mechanism is used to locate and retrieve user/IoT device platform hashes stored on local BC. After successful authentication, the global smart contract authorizes users based on the validation of delegation policies stored on BC. In the proposed framework, global BC is used to validate and store delegation policies. Therefore, the "delegation policy" creation, revocation, and enforcement are transparent to all the users.

A. OUR CONTRIBUTIONS

- The proposed novel framework performs permission delegation and access control for IoT/user devices within a single domain and across different domains. Moreover, the BC in the proposed mechanism makes "delegation policy" creation, revocation, and enforcement transparent to all the users.
- Platform verification mechanism is implemented in global BC to prevent both adversary and legitimate users from exploiting delegated permission. Therefore, during registration, the smart contract attests user/IoT device platform hash from manufacturer and stores on the BC.
- It replaces single trusted permission delegation and authorization service with BC since it is a single point of failure and vulnerable to different attacks.
- The proposed framework preserves the user's privacy by allowing them to get authentication in their parent domains. A novel PoAI mechanism is used during authentication to find and retrieve user/IoT device platform hashes stored on local BC.

The rest of the paper is organized as follows. In Section II, we discuss and summarize preliminaries and related works. In section III, we present an overview of the proposed xDBAuth framework. In section IV, we present the formal modeling of the xDBAuth framework. In Section V, we provide xDBAuth implementation and evaluation details. In section VI, we perform a security analysis of the xDBAuth framework. In section VII, conclusions are drawn.

II. PRELIMINARIES AND RELATED WORKS

A. BLOCKCHAIN

Satoshi Nakamoto introduces the concept of BC in 2008 [9]. BC is an immutable and distributed ledger of blocks that are chained together by cryptographic hashes [10]. The first block in the chain is the genesis block and it does not contain any block hash. Each block consists of a previous block hash, Proof of Work (PoW), block header, time-stamp, and a group of transactions. The following are the main components of BC.

1) DISTRIBUTED LEDGER

Every BC node keeps a copy of the ledger. This distributed ledger contains the current state of the BC. BC uses replication to synchronized the distributed ledgers.

2) ASYMMETRIC CRYPTOGRAPHY

The BC uses a private key for a digital signature to ensure transaction integrity.

3) CONSENSUS MECHANISM

A consensus mechanism is used in BC to achieve agreement on only one value [11]. In other words, it is used to keep a single state of the distributed ledger among all peers.

4) PEER-TO-PEER NETWORK

A P2P network is the interconnection of computers without a central entity. In the absence of a central entity, the BC system depends on consensus among P2P network nodes to add a new block of transactions to the chain [12].

B. PERMISSIONED AND PERMISSIONLESS BC NETWORKS

The BC networks are divided into public BC, private BC or consortium BC [13]. The public or permissionless BC network is fully decentralized. Anyone can read from BC and participate in the mining process. The public BC relies on the PoW consensus mechanism which requires high computational power. Bitcoin[1] and ethereum[2] are some of the examples of public BC. Similarly, the private or permissioned BC allows members with the required permissions to read or write to the chain. Unlike public BC, private BC requires less computational power to add a block to the chain. Hence, it is faster and more efficient. Hyperledger[3] is an example of private BC. The consortium BC network is partially decentralized because it is controlled by a set of pre-defined peers. The consortium BC provides higher efficiency and transaction privacy than public BC. Some of the examples of consortium BC are R3 Corda[4] and Quorum.[5]

C. SMART CONTRACT

The computer application or program that contains pre-defined rules under which the entities communicate with each other is called smart contract [14]. It was first proposed by Nick Szabo in 1994. The smart contract contains a set of pre-defined primitives written in a programming language like Go, JavaScript, Java, Solidity [15]. The smart contract runs automatically on the top of BC when the rule is met. It eliminates the need for a trusted intermediary like the bank, to transfer any asset having value.

D. BC BASED AUTHENTICATION AND AUTHORIZATION WITHIN A SINGLE DOMAIN

A BC based access control framework has been proposed in [18] for IoT. A single smart contract performs user authorization based on access policies. These access policies are stored in BC. Furthermore, IoT devices are not a part of the BC network. Therefore, they use ''management hub'' to execute the smart contract and access the resources. The smart contract allows the requested resource after successful validation of access policies. Similarly, a BC based access management framework is proposed in [19]. It allows a resource owner to publish an access control policy for his resources on the BC. The BC validates user requests against access control policies stored on BC. Upon successful validation, the user is allowed to access the BC protected resource. Furthermore, a user can further transfer the delegated right to other users. The access control policies and the rights transfers are visible to the users for audit. Similarly, a BC based access control framework for IoT cloud has been presented in [20]. The proposed framework enables users to audit authorization decisions. It integrates BC to develop trust among nodes in the BC network. However, it is assumed that all user/IoT devices are trusted.

[1] [Online]. Available: https://bitcoin.org/
[2] [Online]. Available: https://www.ethereum.org/
[3] [Online]. Available: https://www.hyperledger.org
[4] [Online]. Available: https://www.r3.com/
[5] [Online]. Available: https://www.goquorum.com/

E. CROSS-DOMAINS BC BASED AUTHENTICATION AND AUTHORIZATION

In [21], authors have proposed a BC based authentication and authorization mechanism called VeidBlock for Software Defined Network (SDN). It allows SDN devices to generate BC based verifiable identities using VeidBlock i.e. Verifiable Identity Block. During VeidBlock generation, the SDN device gets authentication from the local registration repository or Domain CA. Then, the Identity Provider (Authority) and Validator (IPV) generate VeidBlock for the SBN device. VeidBlock contains the anonymous identity of the requesting SDN device. Here, domain CA is a single point of failure. Also, the proposed framework did not define the permission delegation mechanism.

In [22], the authors have proposed a lightweight access control framework for IoT. Smart homes, cloud storage, and overlay network are the main components of the proposed framework. The proposed framework has eliminated the concept of PoW and coin due to the limited computing power of IoT devices. However, the authors believe that still the BC security and privacy features are maintained. Furthermore, in [23], the authors have elaborated the core components i.e., smart home, cloud storage, and overlay network of their previously proposed framework. In the proposed scenario, different smart homes are connected together using an overlay network. These smart homes can share resources if allowed by the access policies stored on BC. Similarly, local BC is used to manage devices in a single smart home.

The proposed BlendCAC [24] is a BC based access control framework for IoT. The proposed framework delegates and revokes permissions using capability token. These capability tokens are managed through a smart contract. Similarly, IoT Passport [25] is a cross-platform BC based framework. The collaboration of local and global trust domains form IoT passport framework. The local trust domain shares resources among IoT devices within an organization whereas the global trust domain shares resources among collaborating organizations. Furthermore, the proposed smart contract uses cross-platform access control policies to develop trust among collaborating organizations.

Similarly, BC based access control framework called FairAccess has been proposed in [26]. The FairAccess is a privacy-preserving and decentralized framework for IoT. It allows a resource owner to define access policy for his resource and stores it on the BC. Any user can access and further delegate permissions on a resource if it is allowed in the policy.

F. AUTHENTICATION AND AUTHORIZATION THROUGH A CENTRALIZED TRUSTED THIRD PARTY

A centralized cross-domain authentication and authorization framework has been proposed in [4]. The authors have proposed Delegation Service (DS) i.e., a trusted third party for cross-domain authentication and authorization. The DS redirects the external users to his parent domain for authentication. Upon successful authentication, DS authorizes external user based on access policies defined by SP.

A centralized capability-based access management framework has been proposed in [27]. The authors have defined the capability as an object with allowed permissions. Initially, limited permissions are given to the SR. Upon SR request, additional permissions are delegated in the form of capability. Similarly, IoT-OAS [28] is an authorization framework for IoT. The SP redirects user request for authorization to OAuth based authorization service (OAS). OAS is a centralized trusted entity that validates user requests against access policies provided by SP.

III. OVERVIEW OF THE PROPOSED xDBAuth ARCHITECTURE

The proposed xDBAuth is a decentralized permission delegation and access control framework for IoT. It consists of local and global smart contracts. The local smart contract performs authentication and authorization of internal users within a domain. Similarly, the global smart contract allows external users to get authentication from their parent domain. Then, the external user is authorized on the validation of global delegation policies. In the following sections, we discuss the proposed system architecture, system interactions, and compatible use cases.

A. SYSTEM ARCHITECTURE

The proposed architecture is depicted in **Figure 1**. It consists of five core components.

1) Underlay network or IoT domains.
2) Overlay network or virtual coalition of IoT domains.
3) BC manager.
4) User/IoT devices.
5) Smart contract.

1) UNDERLAY NETWORKS OR IoT DOMAINS

Each underlay network is a peer-to-peer (p2p) network of IoT devices in a particular domain i.e., smart home, smart hospital, etc. Every underlay network is a local BC network that has a single smart contract and local storage. The smart contract manages local BC. It stores delegation policies and platform hashes of the internal users in local BC. Each underlay network is called the IoT domain.

2) OVERLAY NETWORK OR VIRTUAL COALITION OF IoT DOMAINS

Overlay network is a peer-to-peer (p2p) network of nodes. Every domain deploys a node to form an overlay network. For example, an overlay network of smart homes and maintenance companies. Therefore, the overlay network is a global BC network that has a single smart contract and global storage. The global storage stores cross-domain delegation policies for external users. The overlay network forms a virtual coalition of IoT domains. These IoT domains use BC to develop trust in each other.

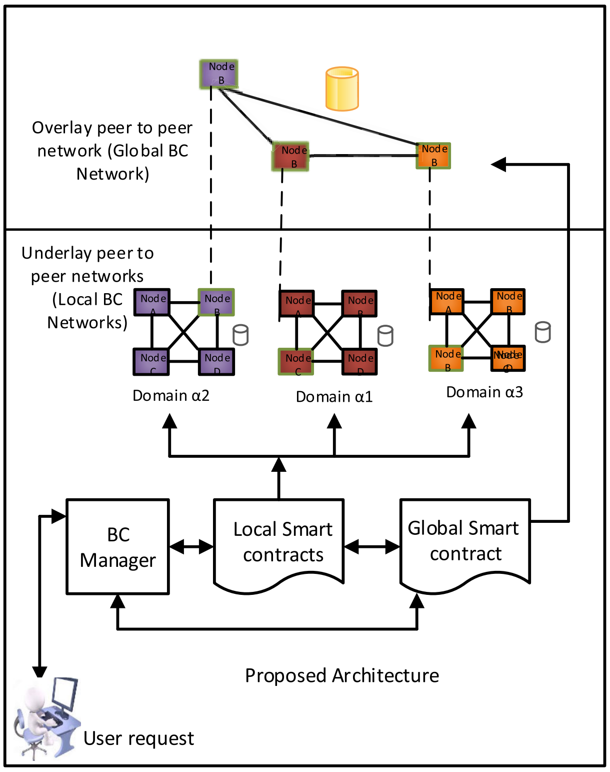


FIGURE 1. High-level xDBAuth Architecture.

3) BC MANAGER

BC manager allows users to access IoT devices/information within a single domain and across domains. When a BC manager receives a request from a user, it generates a BC transaction. The BC transactions are given in **Table 1**. Then, the BC manager forwards the transaction to the smart contract on the BC. We have stored delegation policies in off-chain storage due to the limitation of storage on BC. Therefore, only policy hashes are stored in BC. The details tasks of BC manager are shown in **Figure 2**. The following are the functions of the proposed BC manager.

1) On receiving a registration request, BC manager generates "T.register" transaction to register a node/domain at BC.
2) Allow the owner to publish and revoke delegation policies for his resource using "T.delegate" and "T.revoke" transactions respectively.
3) Generate "T.access" transaction, to access IoT device/information within the same domain or from other domains.

4) USER/IoT DEVICES

The user/IoT devices have unique identities. These identities are used to uniquely identify the user/IoT devices in the local BC network of a particular domain. However, the user device uses his pseudonymous IDs, when accessing cross-domain resources. In the proposed architecture, each user/IoT device must belong to a specific domain. In IoT applications, each device may contain some resources i.e., services and

TABLE 1. BC Transactions.

"T.register"	To register a domain in the global BC network.
"T.access"	To access a resource protected through policies stored at the BC.
"T.delegate"	To create a delegation policy and store at the BC.
"T.revoke"	To revoke a delegation policy from the BC.

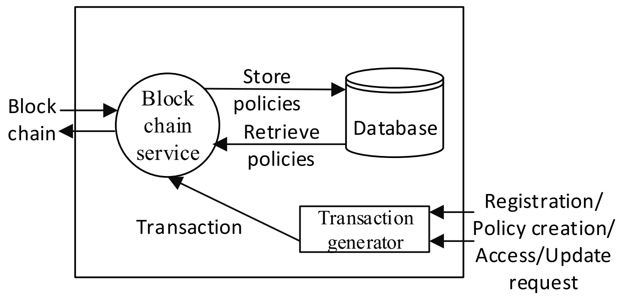


FIGURE 2. BC Manager.

information. The domain owner provides these resources to other nodes. We assume that user/IoT devices have Trusted Platform Module (TPM) that performs platform measurement [29], [30].

5) SMART CONTRACT

The smart contract defines all the operations of an access control system. We have proposed two types of smart contract i.e., local and global smart contracts. These local and global smart contracts are deployed in underlay and overlay networks respectively. These smart contracts have defined authentication and authorization operations for both internal and external user/IoT devices. These operations are described in detail in sections III and IV. The smart contract executes these operations when it receives BC transactions from BC manager. The local smart contract stores user/IoT devices platform hashes and local delegation policies in two different data structures. Similarly, the global smart contract stores global delegation policies in his data structure. The data structure that stores platform hashes is shown in **Table 2**. It consists of "Device ID", "Device pseudonymous IDs", and "Device platform hash" fields. Similarly, "Delegator ID", "Delegatee ID" and "permission" are the fields of the data structure that contains delegation policies. The data structure that stores local/global delegation policies is shown in **Table 3**.

B. OUR ASSUMPTION

In our system design, we assume user/IoT devices are equipped with a TPM module. We assume that the user/IoT device can not use the old platform hash because platform hashes are signed and timestamped by the TPM.

TABLE 2. User device Platform Hashes stored at Local Smart Contract.

S.No	Device IDs	Device Pseudonymous IDs	Platform hash
1	8943552348	8794532784	66f2348794
2	2714532944	4945672348	47acbf8c23
3	9874552244	3343554498	63c35a8f72

TABLE 3. Delegation Policies stored at Smart Contract.

S.No	Delegator ID	Delegatee ID	Permission
1	89a35a234c	78f23b8744	54c674a896, read
2	3d145b294a	48a45f8c24	724c64f736, read/write
3	58a45b2f44	33c35a8f74	89a6f4c296, access

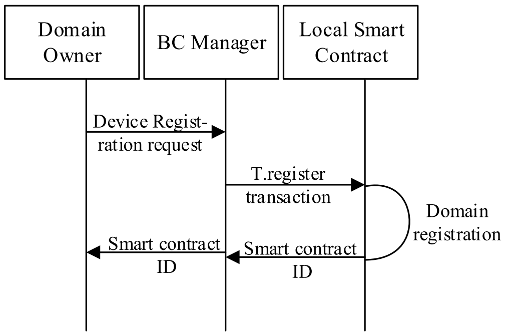


FIGURE 3. Domain Registration Implemented in xDBAuth.

C. SYSTEM OPERATIONS

The following are the different operations perform by the proposed framework.

1) Domain registration in the global smart contract.
2) User/IoT devices registration in the local smart contract.
3) "Delegation policy" publication.
4) Resource access procedure.

1) DOMAIN REGISTRATION IN GLOBAL SMART CONTRACT

The virtual coalition of the IoT domains consists of SP domains and SR domains e.g., a virtual coalition of smart homes and maintenance companies. To register the SP domain in the virtual coalition, the administrator sends a registration request to the BC manager. The registration request contains information like Domain name, IP address, and other meta-data. The BC manager generates "T.register" transaction. Then, it sends the transaction to the global smart contract. Similarly, the administrator of the SR domain registers his domain on the BC. After successful registration, the BC returns a smart contract address. The domain registration in BC is shown in **Figure 3**.

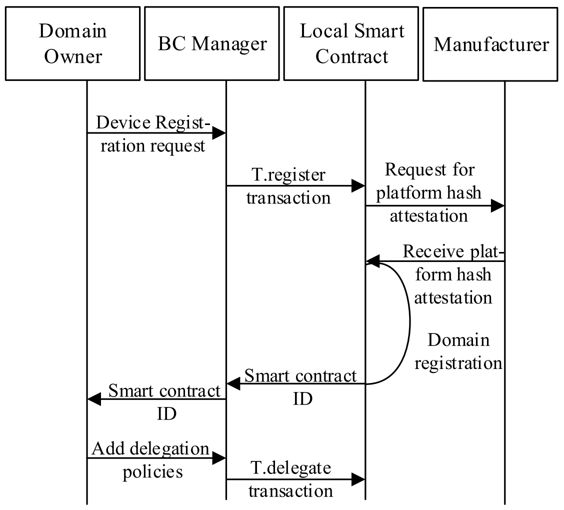


FIGURE 4. User/IoT Device Registration Implemented in xDBAuth.

2) USER/IoT DEVICES REGISTRATION IN LOCAL SMART CONTRACT

We assume that the user/IoT devices have TPM installed. The user/IoT device generates public/private keys using TPM. Similarly, it generates pseudonymous IDs by applying a hash function on his public key. The user/IoT device needs to register his pseudonymous IDs before usage with BC. The user/IoT devices registration in a local smart contract is shown in **Figure 4**. The following are the steps in user/IoT device registration in a local smart contract.

- In step 1, a user/IoT device sends a registration request that consists of public key and platform hash to the BC manager.
- In step 2, the BC manager generates a "T.register" transaction and sends it to the local smart contract.
- In step 3, the local smart contract sends a user/IoT device attestation request to the manufacturer.
- In step 4, the manufacturer attests the platform hash of the device.
- In step 5, the local smart contract registers the user/IoT device. During registration, the local smart contract binds user/IoT device public key and pseudonymous IDs with his platform hash and stores a copy in the local BC.
- In steps 6 and 7, at the end of successful user/IoT device registration, local smart contract sends smart contract address through BC manager to the user/IoT device.
- In steps 8 and 9, the owner/administrator publishes delegation policies for his resource on the BC using BC manager.

3) "DELEGATION POLICY" PUBLICATION

The "Delegation policy" publication for user/IoT devices is shown in **Figure 5**.

- In step 1, administrator/owner unicasts delegation policy request to the BC manager. Then, the BC manager generates "T.delegate" transaction.
- In step 2, BC manager sends "T.delegate" to the local smart contract.

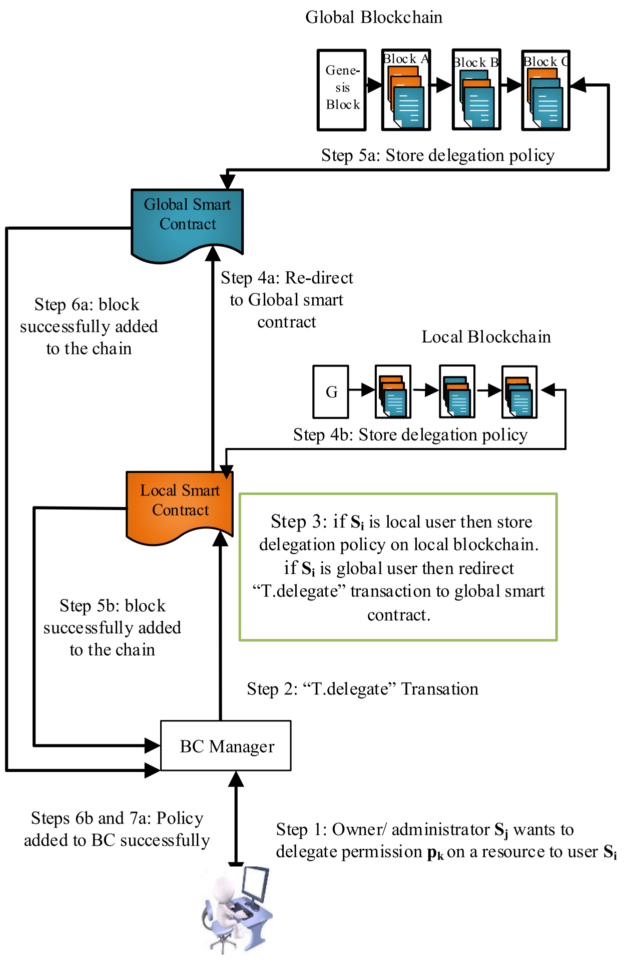

Global Blockchain
Genesis Block
Block A
Block B
Block C
Step 5a: Store delegation policy
Global Smart Contract
Step 6a: block successfully added to the chain
Step 4a: Re-direct to Global smart contract
Local Blockchain
G
Step 4b: Store delegation policy
Local Smart Contract
Step 3: if $\mathbf{S_i}$ is local user then store delegation policy on local blockchain. if $\mathbf{S_i}$ is global user then redirect "T.delegate" transaction to global smart contract.
Step 5b: block successfully added to the chain
Step 2: "T.delegate" Transation
BC Manager
Steps 6b and 7a: Policy added to BC successfully
Step 1: Owner/ administrator $\mathbf{S_j}$ wants to delegate permission $\mathbf{p_k}$ on a resource to user $\mathbf{S_i}$

FIGURE 5. "Delegation policy" publication Implemented in xDBAuth.

- In step 3, the local smart contract determines whether the delegatee s_i is internal or external user.
- In step 4a, "T.delegate" transaction is redirected to global smart contract when s_i is external user.
- In step 4b, the delegation policy is stored on local BC when s_i is internal user.
- In step 5a, the delegation policy is stored on global BC.
- In step 5b, the local smart contract sends "block successfully added to the chain" message to the BC manager.
- In step 6a, the global smart contract sends "block successfully added to the chain" message to the BC manager.
- In step 6b and 7a, the BC manager sends "delegation policy successfully added to the chain" message to the administrator/owner.

4) RESOURCE ACCESS PROCEDURE

The resource access procedure for both external and internal user/IoT devices is shown in **Figure 6**. It consists of the following steps.

- In step 1, internal/external user unicasts an access request to the BC manager. Then, the BC manager generates a "T.access" transaction.

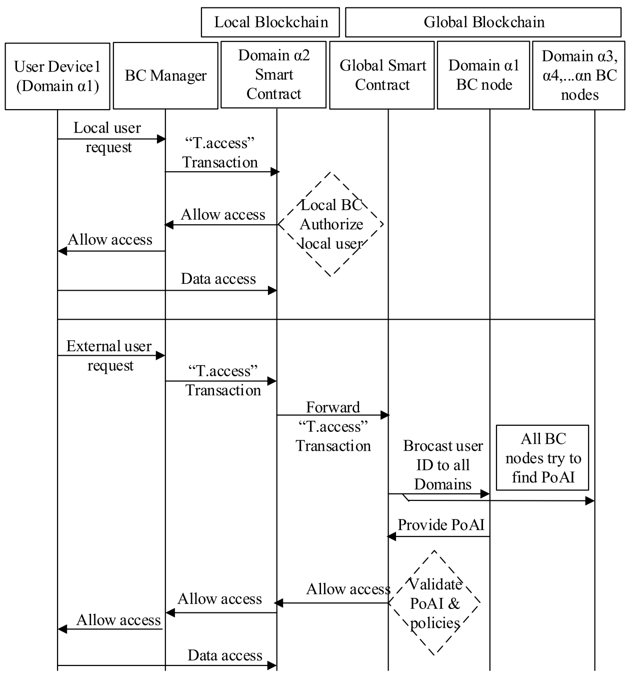


FIGURE 6. Resource Access procedure Implemented in xDBAuth.

- In step 2, BC manager sends a "T.access" transaction to the local smart contract on the BC. The local smart contract authenticates and authorizes internal users based on platform hashes and local delegation policies respectively.
- In step 3a, the local smart contract redirects the external user's request for authentication and authorization to the global smart contract.
- In step 3b, the local smart contract sends allow/deny message to the requester after platform hashes and local delegation policies validation.
- In steps 4a and 4b, the global smart contract broadcasts the external user/IoT device pseudonymous ID to all the peers in the global BC network for authentication. Each peer in the global BC network belongs to different IoT domains. Every peer tries to find PoAI for the requesting user/IoT device in the local BC. PoAI is a mechanism that finds platform hash for a given pseudonymous ID of a user/IoT device. However, only the user's parent domain, who has registered the user/IoT device, will be able to find its platform hash. The parent domain returns the PoAI to the global smart contract.
- In step 5, the global smart contract compares the platform hash received from the user's parent domain with the hash received in the user's request.
- In step 6, if both the hashes are matched, then the global smart contract retrieves delegation policies from global BC.
- In step 7, the global smart contract validates the user's request against the delegation policy.
- In step 8, the global smart contract sends an allow/deny response to the local BC.
- In step 9, local BC allows/denies user's request based on the authorization decision of the global BC.
- In step 10, the BC manager informs the requesting user about the authorization decision.

D. COMPATIBLE USE CASES

1) SMART HOME

A smart home technology also called home automation or domotics is uses to control devices automatically, such as lighting and heating in a residence. These devices can stop working at any time because they work 24/7 hours. Suppose an owner of a smart homeowner has a maintenance contract with a company. Therefore, he calls to the company for maintenance from his office. Traditionally, the maintenance technician gets the owner's username and password and login into the smart home controlling system. In this scenario, the maintenance technician got full access permissions of all the devices. He can steal important data or install the malware on the devices. A trivial solution is to create a new account for every external maintenance technician and assign proper permissions. This leads to a situation where the system has to handle many external accounts. Alternatively, an owner can explicitly delegate corresponding permissions to the maintenance technician. Resource access procedure for maintenance technician is shown in **Figure 7.** The following steps describe a resource access procedure for maintenance technicians.

- In step 0a, the administrator/owner unicasts a delegation policy request to the BC manager. Then, the BC manager generates a "T.delegate" transaction.
- In step 0b, BC manager sends "T.delegate" to the global smart contract.
- In step 0c, the delegation policy is stored on global BC.
- In step 1, the technician unicasts an access request to the BC manager. Then, the BC manager generates a "T.access" transaction.
- In step 2, BC manager sends a "T.access" transaction to the smart contract of the smart home α_1.
- In step 3a, the smart contract redirects the technician request for authentication and authorization to the global smart contract.
- In steps 4a and 4b, the global smart contract broadcasts the maintenance technician's pseudonymous ID to all the peers i.e., maintenance companies and smart homes in the global BC network for authentication. Every peer tries to find platform hash for technician pseudonymous ID. However, only the technician's parent domain, who has registered the user/IoT device, will be able to find its platform hash. The parent domain returns the PoAI to the global smart contract.
- In step 5, the global smart contract compares the platform hash received from the technician's parent domain with the hash received in the technician request.
- In step 6, if both the hashes are matched, then the global smart contract retrieves delegation policies from global BC.

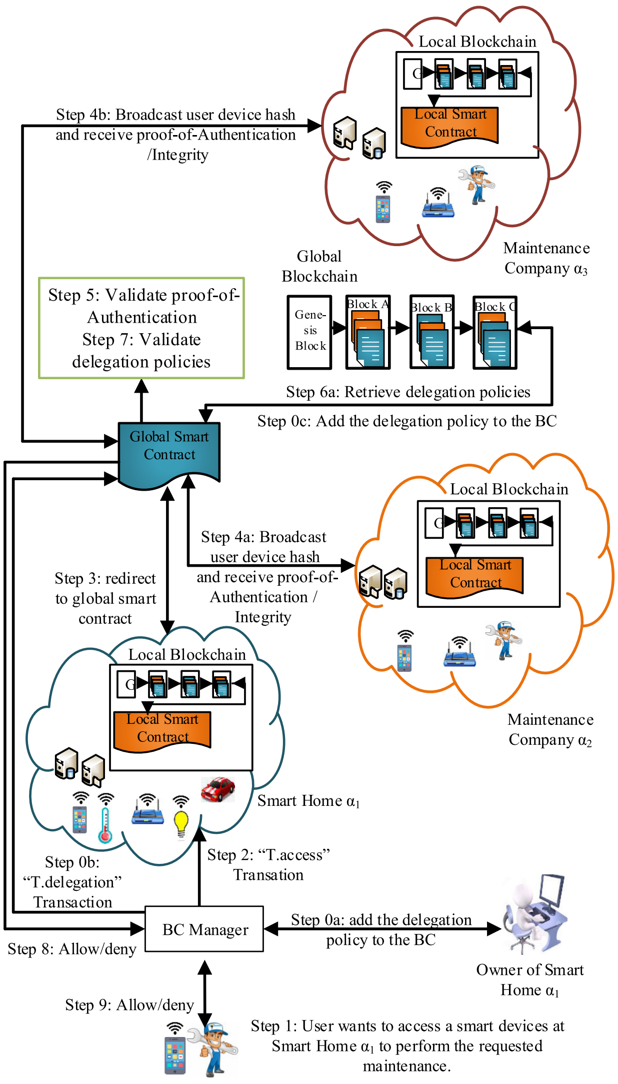

Local Blockchain
Local Smart Contract
Step 4b: Broadcast user device hash and receive proof-of-Authentication /Integrity
Global Blockchain
Maintenance Company α3
Step 5: Validate proof-of-Authentication
Step 7: Validate delegation policies
Genesis Block
Block A
Block B
Block C
Step 6a: Retrieve delegation policies
Step 0c: Add the delegation policy to the BC
Global Smart Contract
Local Blockchain
Step 4a: Broadcast user device hash and receive proof-of-Authentication / Integrity
Step 3: redirect to global smart contract
Local Blockchain
Local Smart Contract
Local Smart Contract
Maintenance Company α2
Smart Home α1
Step 0b: "T.delegation" Transaction
Step 2: "T.access" Transation
BC Manager
Step 0a: add the delegation policy to the BC
Step 8: Allow/deny
Owner of Smart Home α1
Step 9: Allow/deny
Step 1: User wants to access a smart devices at Smart Home α1 to perform the requested maintenance.

FIGURE 7. Resource Access procedure for Maintenance Technician.

- In step 7, the global smart contract validates the user's request against the delegation policies.
- In step 8, the global smart contract sends an allow/deny response to the BC manager.
- In step 9, the BC manager informs the requesting user about the authorization decision.

2) SMART HEALTH

With the advance in transportation, patient mobility increases. Therefore, Electronic Medical Records (EMRs) must be made available to different healthcare providers within and across the national borders. National Health Information Network (NHIN) architecture, is a virtual coalition of multiple health care organizations. They share patients EMRs. EMR contains patient medical and clinical data. Also, they provide permissions to the external physicians on different devices and sensors attached to the patient body. For example, a physician in hospital **A** wants to discuss a patient disease with a physician in hospital **B**. The physician in hospital **B** needs permissions on all the devices and sensors attached to the patient body to diagnose the patient. Therefore, the hospital **A** access control system authenticates and provides the required permissions to the external physician. Traditionally, both authentication and authorization are performed at the resource owner domain or by a trusted third party. Also, it is a single point of failure. Therefore, a framework is required that authenticates the external user in her parent domain. Then, the external user is authorized to access the resource based on the BC consensus.

TABLE 4. Notations.

Notation	Description
s_i	ith subject.
o_j	jth object.
p_k	kth permission.
att_x	xth attribute.
D_i	ith Domain.
$\mathcal{VC}$	virtual coalition of domains.
$\hbar_{s_i,status}$	platform hash of s_i where status = c (current) or s (stored) or r (received)
$\{s_i \rightsquigarrow^{p_k} s_j\}$	s_i delegated a permission p_k to s_j.
$\mathcal{DS_L}$	a super set of all local delegations
$\mathcal{DS_G}$	a super set of all cross-domain delegations
$\mathcal{DS_{HS}}$	a super set of all user/IoT devices platform hashes

IV. FORMAL MODELING OF THE PROPOSED ARCHITECTURE

In this section, we formally define main components of the proposed architecture. Then, the main functions i.e., "delegation policy" creation and revocation, local and global smart contracts operations are discussed in detail.

A. NOTATIONS

The notations use in the proposed work are described in **Table 4**.

B. DEFINITIONS

1) SUBJECT SET

$\mathcal{S} \supseteq \cup_{i=0}^{\infty} s_i$, $\mathcal{S}$ is a superset of all the persons or things that make a request for a resource in IoT. A subject acts both as a delegator and delegatee. A delegator is a person or thing that gives permission on a resource. Similarly, delegatee is a person or thing that receives the permission on a resource. User Set (US) $\subseteq \mathcal{S}$, Where US contains human operators and user/IoT devices. The US is further divided into Local User Set (LUS) and Global User Set (GUS). LUS contains all the internal users of a domain Whereas, GUS contains external users that belong to different domains.

2) OBJECT SET

$\mathcal{O} \supseteq \cup_{j=0}^{\infty} o_j$, $\mathcal{O}$ is a superset of all the things.i.e., data, service that are requested by an user in IoT.

3) PERMISSION SET

$\mathcal{P} \supseteq \cup_{j=0}^{\infty} p_k$, $\mathcal{P}$ is a superset of all the permissions in IoT. Permission p_i is the ith pair of (o_j,a), where o_j is object and **a** is an action. In other words, permission represents action on an object.

4) ATTRIBUTE SET

$\mathcal{ATT} \supseteq \cup_{x=0}^{\infty} att_x$. $\mathcal{ATT}$ contains all the attributes of all the subjects in the system. The att_x is the xth attribute in $\mathcal{ATT}$. Attribute is a characteristic of a subject. Formally, the subject with associated attribute can be written as $(s_i.att_x)$. For example, if s_i is the administrator/owner of the resource then, we write attribute as (s_i.admin).

5) DELEGATION

We define delegation as a triple (s_i, s_j, p_k), where s_i is a delegator, s_j is a delegatee, and p_k is a permission or set of delegated permission. $(s_i, s_j, p_k) \in s_i \leadsto^{p_k} s_j$

6) IoT DOMAIN

$\mathcal{D}_i \supseteq \cup_{i=0,j=0,k=0}^{\infty} \{s_i \cup o_j \cup p_k \cup \mathcal{DS}_{\mathcal{L}}\}$. $\mathcal{D}_i$ is superset of all the subjects, objects and permissions in a domain.

7) VIRTUAL COALITION

$\mathcal{VC} \supseteq \cup_{i=2}^{\infty} D_i$, such that $\{s_i \in D_i\}$, $\{o_j \in D_j\}$ and $\{s_i \leadsto^{p_k} s_j\} \in \mathcal{DS}_{\mathcal{G}}$. $\mathcal{VC}$ is a superset of all the IoT domains. Virtual coalition is the combination of two or more IoT domains such that device in one domain is allowed to access a resource in another domain.

8) PSEUDONYMOUS IDs

$\mathcal{PID} \supseteq \cup_{i=0}^{\infty} \mathcal{PID}_i$. $\mathcal{PID}_i$ is super set of all the pseudonymous IDs in a domain.

9) LOCAL DELEGATION SET

$\mathcal{DS}_{\mathcal{L}} \supseteq \cup_{i=0,j=0,p=0}^{\infty} \{s_i \leadsto^{p_k} s_j\}$, where $\mathcal{DS}_{\mathcal{L}}$ is a super set of all the delegation policies created for local user within domain.

10) GLOBAL DELEGATION SET

$\mathcal{DS}_{\mathcal{G}} \supseteq \cup_{i=0,j=0,p=0}^{\infty} \{s_i \leadsto^{p_k} s_j\}$, where $\mathcal{DS}_{\mathcal{G}}$ is a super set of all the delegation policies created for external user of other domains.

11) PLATFORM HASHES SET

$\mathcal{HS} \supseteq \cup_{i=0,status=c,s,r}^{\infty} \{\hbar_{s_i,status}\}$, where $\mathcal{HS}$ is a super set of the platform hashes of all the user/IoT devices s_i within a domain.

12) "T.register" TRANSACTION

It is a cartesian product of a user, user attribute i.e., user must be admin/owner, and device platform hash value. Formally, $\{\mathcal{S} \times (\mathcal{S}.\mathcal{ATT} == admin/owner) \times \mathcal{HS}\}$.

13) "T.delegate" TRANSACTION

It is a cartesian product of delegator, delegatee, permission, and delegator attribute i.e., delegator must be admin/owner. Formally, $\{\mathcal{S} \times \mathcal{S} \times \mathcal{P} \times (\mathcal{S}.\mathcal{ATT} == admin/owner)\}$.

14) "T.access" TRANSACTION

It is a cartesian product of delegator, delegatee and permission. Formally, $\{\mathcal{S} \times \mathcal{S} \times \mathcal{P} \times \mathcal{HS}\}$.

15) "T.revoke" TRANSACTION

It is a cartesian product of delegator, delegatee, permission, and delegator attribute. Formally, $\{\mathcal{S} \times \mathcal{S} \times \mathcal{P} \times (\mathcal{S}.\mathcal{ATT} == admin/owner)\}$.

C. DOMAIN REGISTRATION

$\overline{\mathcal{VC}} = \mathcal{VC} + D_i$ Where $D_i = \cup_{i=0,j=0,k=0}^{\infty} \{s_i \cup o_j \cup p_k \cup \{s_i \leadsto^{p_k} s_j\}\}$. s_i is an external user, o_j is a resource allowed to external user, p_k is the permission on the object, and $\{s_i \leadsto^{p_k} s_j\}$ is a set of global policies for the external user s_i.

D. USER/IoT DEVICE REGISTRATION

During registration, the new user/IoT device s_i is added to $\overline{LUS} = LUS + s_i$, where s_i is the user/IoT device public key. Similarly, user/IoT device platform hash is attested from the manufacturer and added to $\overline{\mathcal{HS}} = \mathcal{HS} + \hbar_{s_i}$, where $\hbar_{s_i}$ is the platform hash of user/IoT device s_i.

E. "DELEGATION POLICY" CREATION

The **Algorithm 1**, takes delegation policy request i.e., "T.delegate" transaction as input and stores it on local/global BC. **Algorithm 1** returns an error if the delegation policy already presents on the BC or delegatee is not the owner of the resource. In lines 3-8, if the delegator in the policy is the owner/administrator of the resource, delegatee is an internal user, and a similar delegation policy does not exist on the local BC. Then, **Algorithm 1** adds the delegation policy on the local BC. Similarly, in lines 9-14, **Algorithm 1** adds a delegation policy for the external user in the BC if the delegator in the policy is the owner/administrator of the resource, delegatee is an external user, and similar delegation policy does not exist on the global BC. In lines 15-16, **Algorithm 1** returns an error, if any condition is false in the above given conditions. The "Delegation Policy" creation algorithm is given in **Algorithm 1**.

F. "DELEGATION POLICY" REVOCATION

The proposed mechanism allows only the administrator/owner to revoke both local and global delegation policies from internal and external users respectively. The "delegation policy" revocation algorithm is given in **Algorithm 2**. **Algorithm 2** takes "delegation policy" revocation request i.e., "T.revoke" transaction as input and removes the policy on local/global BC. **Algorithm 2** returns an error if delegation policy does not exist on the BC or delegatee is not the owner of the resource. In lines 3-5, if the delegator in

Algorithm 1 "Delegation Policy" Creation

```
1:  Input: T.delegate($s_i$,$s_j$,$p_k$, $s_i.att_x$)
2:  Output: ($s_i \rightsquigarrow^{p_k} s_j$) or error
3:  if ($s_i.att_x$ == admin/owner ) && ($s_j \in LUS$) then
4:      if $\{s_i \rightsquigarrow^{p_k} s_j\} \in \mathcal{DS_L}$ then
5:          return error (Delegation already exist)
6:      else
7:          $\overline{\mathcal{DS_L}} = \mathcal{DS_L} + \{s_i \rightsquigarrow^{p_k} s_j\}$
8:      end if
9:  else if ($s_i.att_x$ == admin/owner ) && ($s_j \in GUS$) then
10:     if $\{s_i \rightsquigarrow^{p_k} s_j\} \in \mathcal{DS_G}$ then
11:         return error (Delegation already exist)
12:     else
13:         $\overline{\mathcal{DS_G}} = \mathcal{DS_G} + \{s_i \rightsquigarrow^{p_k} s_j\}$
14:     end if
15: else
16:     return error
17: end if
```

the policy is the owner/administrator of the resource, delegatee is an internal user, and the delegation policy exists on the local BC. Then, **Algorithm 2** removes the delegation policy from the local BC. In lines 6-8, **Algorithm 2** returns an error, if the delegator in the local policy is not the owner/administrator. Similarly, in lines 9-11, **Algorithm 2** removes a delegation policy of an external user from the BC if the delegator in the policy is the owner/administrator of the resource, delegatee is an external user, and the delegation policy exists on the BC. In lines 12-14, **Algorithm 2** returns an error, if the delegator in the global policy is not the owner/administrator.

Algorithm 2 "Delegation Policy" Revocation

```
1:  Input: T.revoke($s_i$,$s_j$,$p_k$, $s_i.att_x$)
2:  Output: $\mathcal{DS_L}$ - $\{s_i \rightsquigarrow^{p_k} s_j\}$ OR $\mathcal{DS_G}$ - $\{s_i \rightsquigarrow^{p_k} s_j\}$
3:  if ($s_i.att_x$ == admin/owner ) && ($s_j \in LUS$) then
4:      if $\{s_i \rightsquigarrow^{p_k} s_j\} \in \mathcal{DS_L}$ then
5:          $\overline{\mathcal{DS_L}} = \mathcal{DS_L} - \{s_i \rightsquigarrow^{p_k} s_j\}$
6:      else if $\{s_i \rightsquigarrow^{p_k} s_j\} \notin \mathcal{DS_L}$ then
7:          return error
8:      end if
9:  else if ($s_i.att_x == admin/owner$) && ($s_j \in GUS$) then
10:     if $\{s_i \rightsquigarrow^{p_k} s_j\} \in \mathcal{DS_G}$ then
11:         $\overline{\mathcal{DS_G}} = \mathcal{DS_G} - \{s_i \rightsquigarrow^{p_k} s_j\}$
12:     else if $\{s_i \rightsquigarrow^{p_k} s_j\} \notin \mathcal{DS_G}$ then
13:         return error
14:     end if
15: end if
```

G. RESOURCE ACCESS PROCEDURE THROUGH LOCAL SMART CONTRACT

Algorithm 3 defines a resource access procedure for internal users. **Algorithm 3** takes access request i.e., "T.access" transaction as input and either allows or denies the user request. In line 3, if the requester is an internal user then the smart contract retrieves the user's stored platform hash from the BC. In lines 4-5, the smart contract compares the user device platform hash received in the user request with the platform hash retrieved from local BC. In line 6, if both hashes match, then it validates the delegation policies. In lines 7-8, the user request is allowed if there is a delegation policy for the user on the BC. In lines 9-10, the user is denied if a delegation policy does not exist on the BC. In lines 12-13, the user is denied when the platform hashes do not match. In line 15-17, the smart contract redirects the external user to the global smart contract.

Algorithm 3 Resource Access Procedure Through Local Smart Contract

```
1:  Input: T.access ($s_j$, $p_k$, $\hbar_c$)
2:  Output: allow or deny
3:  if ($s_j \in LUS$) then
4:      retrieve platform hash from BC.
5:      if ($\hbar_c \equiv \hbar_s$) then
6:          retrieve delegation policy from BC.
7:          if $\{s_k \rightsquigarrow^{p_k} s_j\} \in \mathcal{DS_L}$ then
8:              allow
9:          else if $\{s_k \rightsquigarrow^{p_k} s_j\} \notin \mathcal{DS_L}$ then
10:             deny
11:         end if
12:     else if ($\hbar_c \neq \hbar_s$) then
13:         deny
14:     end if
15: else if ($s_j \in GUS$) then
16:     redirect T.access ($s_j$, $p_k$, $\hbar_c$) to global smart contract
        (Algorithm 4).
17: end if
```

H. CROSS-DOMAIN RESOURCE ACCESS PROCEDURE THROUGH GLOBAL SMART CONTRACT

Algorithm 4 defines the resource access procedure for external users. **Algorithm 4** takes an access request which consists of the pseudonymous ID of the external user device $\mathcal{PID}_{s_j}$, permission p_k on the resource, and external user device platform hash $\hbar_c$ as input and either allows or denies the user request. In line 3, the global smart contract broadcasts user device pseudonymous ID $\mathcal{PID}_{s_j}$ to all the nodes in global BC i.e., each node belongs to a single domain in the virtual coalition. In line 4, the smart contract receives the user device platform hash from the user's parent domain node. In lines 5-9, the smart contract compares the user's device platform hash received in the user request with the platform hash received from the user's parent domain. If both hashes match, then it searches for a valid delegation policy. The user request is allowed when there is a valid delegation policy present in the BC for the requesting user. In lines 10-15, the user is denied when a delegation policy does not exist on the BC or the platform hashes do not match.

Algorithm 4 Cross-Domain Resource Access Procedure Through Global Smart Contract

```
1:  Input: T.access ($\mathcal{PID}_{s_j}, p_k, \hbar_c$)
2:  Output: allow or deny
3:  Broadcast the requester pseudonymous ID ($\mathcal{PID}_{s_j}$) to all
    the nodes of the global BC.
4:  Receive PoAI i.e., platform hash value ($\hbar_s$) of the
    user/IoT device platform.
5:  while (got PoAI from the BC) do
6:      if ($\hbar_c \equiv \hbar_s$) then
7:          retrieve delegation policy from BC.
8:          if $\{s_k \leadsto^{p_k} \mathcal{PID}_{s_j}\} \in \mathcal{DS}_\mathcal{G}$ then
9:              allow
10:         else if $\{s_k \leadsto^{p_k} \mathcal{PID}_{s_j}\} \notin \mathcal{DS}_\mathcal{G}$ then
11:             deny
12:         end if
13:     else if ($\hbar_c \neq \hbar_s$) then
14:         deny
15:     end if
16: end while
```

I. PROOF-OF-AUTHENTICITY/INTEGRITY

The determination of user device platform hash against his pseudonymous ID by the BC is called PoAI. Each local BC runs **Algorithm 5** when they receive a broadcast request from the global smart contract. **Algorithm 5** takes user device pseudonymous ID $\mathcal{PID}_{s_j}$ as input and returns user device platform hash $\hbar_s$ as an output. In line 1, a node in the global BC receives the pseudonymous ID $\mathcal{PID}_{s_j}$ of the user device. In line 2, the local smart contract searches user/IoT device platform hash using the pseudonymous ID in a local BC. In lines 5-6, the node finds the user/IoT device platform hash and returns the hash to the global smart contract. In lines 7-8, the node does not find the user/IoT device platform hash and returns an error.

Algorithm 5 Proof-of-Authenticity/Integrity

```
1: Input: $\mathcal{PID}_{s_j}$
2: Output: $\hbar_s$
3: Receive user device pseudonymous ID $\mathcal{PID}_{s_j}$ in a broad-
   cast request from the global smart contract.
4: Searching for user device platform hash using the
   pseudonymous ID in a local database.
5: if ($\hbar_s$ exist for $\mathcal{PID}_{s_j}$) then
6:     return $\hbar_s$
7: else if ($\hbar_s$ does not exist for $\mathcal{PID}_{s_j}$) then
8:     return error
9: end if
```

V. IMPLEMENTATION DETAILS

We have implemented our proposed framework in Node.js. Our implementation consists of a client application, server node, database node, chaincode, and BC as shown in **Figure 8**.

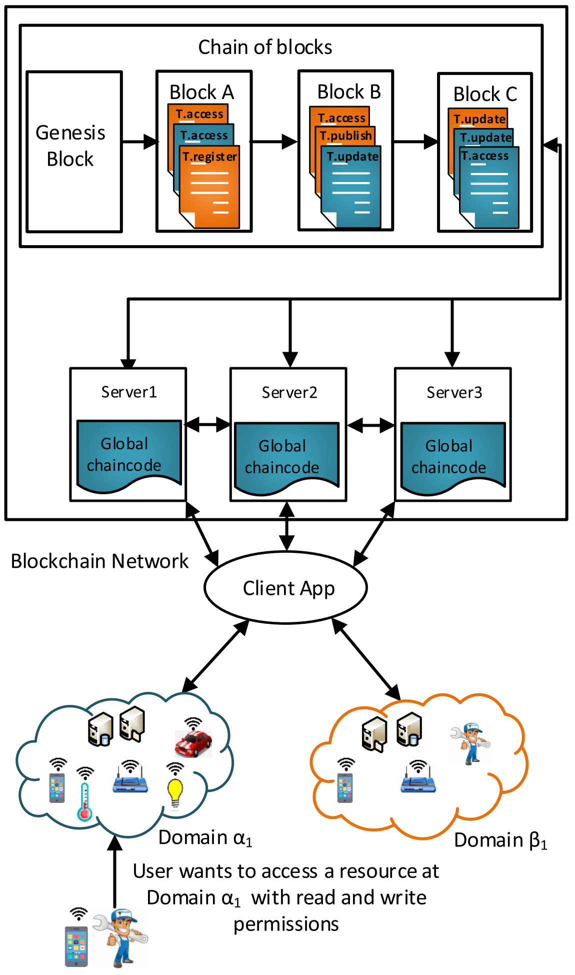


FIGURE 8. xDBAuth Implementation.

1) CLIENT APPLICATION

A client application is used to send and retrieve information from a smart contract. We have used a third party application called postman to send a request and receive a response from the smart contract.

2) SERVER NODE

The server node listens to the client requests on a particular port and calls the chaincode. Similarly, it connects MongoDB with the chaincode.

3) DATABASE NODE

Database node deploys MongoDB. MongoDB holds user delegation policies, user/IoT device platform hashes, and user/IoT device registration.

4) CHAINCODE

A chaincode, also called smart contact, implements business logic. We have developed a global chaincode in node.js to manage global BC. Global chaincode authenticates and authorizes external users. Global chaincode broadcasts external user pseudonymous ID to all IoT domains. In response,

TABLE 5. Block Structure.

perviousHash:	"dfbd24503452ef6dff8eb1a9a87c6ba978 037c268166d89d1f0603fd38f9e861"
delegator:	"Alice"
delegatee:	"Bob"
DeviceId:	"device234"
validTill:	"18-Nov-2019"
hash:	"ac8d393bea9d457d2bcb26c5cfa893189 4ab6d427c8c98b47a55fe8f03cc4cb7"
timestamp:	"Tue, 12 Nov 2019 09:42:57 GMT"

TABLE 6. Network configuration.

Parameters	Values
Database	MongoDB
Node Resources	Intel(R) Core(TM) i5-8350U CPU @ 1.70 GHz with 16 GB RAM
Block Size	571 byte
Network Links	100 Mbps

the global chaincode receives PoAI from the user's parent domain. Then, it compares the user platform hash received in request with the hash received in the PoAI. The user is authenticated only if both the hashes are equal. Upon successful authentication, the global chaincode validates the external user request against the delegation policies stored on BC.

5) BLOCKCHAIN

We have developed our own chain. The chain starts with a genesis block and every subsequent block contains a hash of the previous block in the chain. Our implemented block consists of the following components. The block structure is shown in **Table 5**.

A. NETWORK SETUP

Our test network is shown in **Figure 8**. It consists of 2 IoT domains, 3 server nodes, 2 database nodes. The experiments are performed on two laptops having intel(R) Core(TM) i5-8350U CPU @ 1.70 GHz and 16 GB RAM DDR4. The network configuration is described in **Table 6**.

B. PERFORMANCE EVALUATION AND COMPARISON

In this section, we performed the blockchain size analysis, throughput analysis, computational cost analysis, and overhead ratio analysis of the proposed framework.

1) BLOCKCHAIN SIZE ANALYSIS

We tested the performance of the proposed framework by measuring the execution time of each stage i.e., authentication, authorization, policy creation, and policy revocation based on the proposed scenario. In the experiment, 200 virtual clients (N=200) concurrently send an access request to the global smart contract as shown in **Figure 8**. The total number

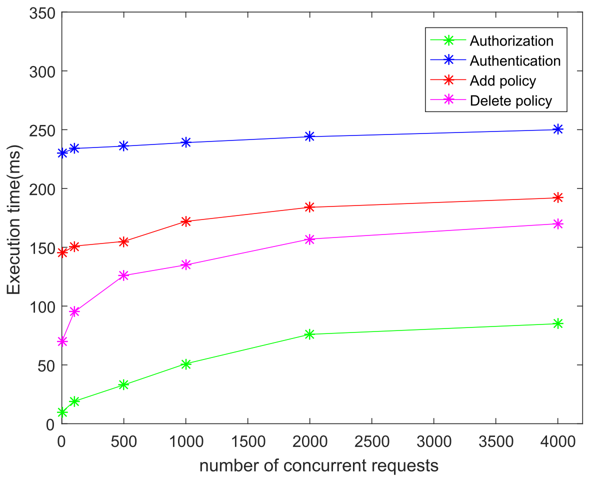


FIGURE 9. Blockchain Size Analysis.

of virtual clients is kept constant. Furthermore, the SPs define delegation policies for their resources and store them in BC. The BC size increases with the increase in the number of delegation policies. Therefore, we evaluated the performance of each stage with 1, 100, 500, 1000, 2000 and 4000 delegation policies. However, the limited scalability of MongoDB restricts the total number of delegation policies. Initially, a BC having 1 delegation policy (p = 1) is tested. The average authentication time takes 224 ms, authorization time takes 20 ms, policy creation time takes 145 ms, and policy revocation time takes 130 ms. Then, the experiment is repeated for BCs that store P= 100, 500, 1000, 2000, and 4000 number of delegation policies. The results, given in **Figure 9**, show that the execution time of user authentication and policy creation is not affected much by the increase in the number of policies. However, user authorization time and policy revocation time increased due to the increase in time required for searching a specific policy in a long chain of policies.

2) THROUGHPUT ANALYSIS

We evaluate the user authentication, authorization, "delegation policy" creation, and revocation operations of the global smart contract with concurrent access requests from N = 50, 200, 400, 600, and 800 virtual clients. N shows the total number of virtual clients. The total number of delegation policies were kept constant i.e., p=1000. Initially, 50 concurrent virtual clients requests were tested. The average authentication time takes 240ms, authorization time takes 30ms, "delegation policy" creation time takes 120ms, and "delegation policy" revocation time takes 87ms. Then, the experiment is repeated for n = 200, 400, 600, and 800. The results are shown in **Figure 10**.

We calculate the trend of average execution time of user authentication, authorization, "'delegation policy" creation, and revocation operations with concurrent access requests from N = 50, 200, 400, 600 and 800 virtual clients. The results show that the throughput of the proposed framework increases with the increase in the number of concurrent

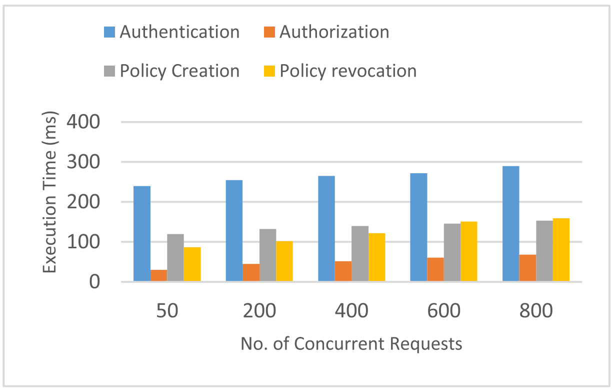


FIGURE 10. Average Execution Times of the Smart Contract Operations.

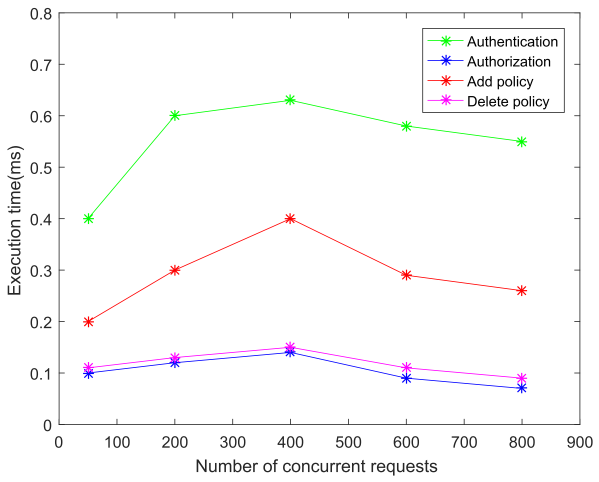


FIGURE 11. Trend of Average Execution Times of the Smart Contract Operations.

requests. However, after a certain level, the throughput tends to be stable. Further increase in the number of requests does not show any significant decrease in throughput. The results are shown in **Figure 11**.

3) COMPUTATIONAL COST ANALYSIS

The proposed xDBAuth session contains several security operations i.e., a hashing function, encryption, decryption, signature generation, and signature verification. The SHA-256 hashing function is used to generate 64 bits block hash which takes 0.87ms. Similarly, Node.js built-in library called ''crypto'' is used to encrypt/decrypt BC transactions. The ''crypto'' uses AES (Advanced Encryption System) encryption algorithm. The BC transactions encryption/decryption take 0.90ms/1.5ms. Furthermore, the signature generation and verification takes 3.95ms and 2ms respectively. The results are given in **Table 7**.

4) OVERHEAD RATIO

We implemented the proposed scenario with and without encryption, decryption, and digital signature using xDBAuth. We kept the number of concurrent virtual client's requests (N=100) and total delegation policies (p=100) constant.

TABLE 7. Computation Cost Analysis.

xDBAuth security operations	Average Computational Cost (ms)	Standard Deviation (ms)
Hash function	0.87	0.04
Transaction Encryption	0.90ms	0.05
Transaction Decryption	1.5ms	0.03
Signature Generation	3.95ms	0.3
Signature Verification	2ms	0.04

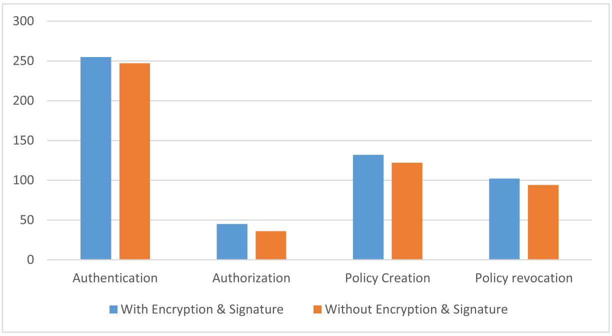


FIGURE 12. Overhead Ratio.

The result is shown in **Figure 12**. The results show that the overhead in all the operations is less than 3%, which is negligible.

The above results show that the average computation costs do not notably increase with the increase in the number of virtual client's requests and delegation policies. Therefore, the proposed framework gives good scalability.

5) COMPARISONS BETWEEN EXITING AND xDBAuth FRAMEWORKS

Comparisons among proposed xDBAuth and existing related frameworks are presented in **Table 8**. The frameworks [4] and [28] have targeted cross-domain access control. However, a single trusted third party performs user authentication and authorization. Here, if the centralized entity fails, then no authentication and authorization will occur. Also, these centralized trusted entities are vulnerable to different attacks like DoS, DDoS, Spoofing, Sybil. Moreover, these centralized trusted entities can leak the user's privacy because it knows the credentials of both SR and SP.

Similarly, the proposed decentralized BC-based frameworks [24] and [26] have targeted user authentication and authorization in cross-domain. However, user/IoT device authentication is not sufficient because even a genuine user/IoT device can misuse delegated permission. Both mechanisms are based on BC therefore, they have high fault tolerance and computation cost. Moreover, both frameworks are vulnerable to DoS, DDoS, Spoofing, and Sybil attacks.

TABLE 8. Comparisons among proposed XDBAuth and Exiting Frameworks.

Access Control Mechanisms	Authentication & Authorization within Single domain	Cross-domain Authentication & Authorization	Cross-domain Delegation	Platform Verification	Privacy	Consensus Mechanisms	Implementations	Decentralization	Fault tolerance	computation cost	DoS/ DDoS Attack	Spoofing Attack	Sybil Attack
[4]	No	Yes	Yes	No	No	No	Single Trusted Server	No	Low	High	Yes	Yes	Yes
[18]	Yes	No	No	No	No	Yes	Ethereum	Yes	High	High	Yes	No	No
[19]	Yes	No	No	No	No	Yes	Bitcoin	Yes	High	High	Yes	No	No
[20]	Yes	No	No	No	No	Yes	Ethereum	Yes	High	High	Yes	No	No
[21]	No	Yes	No	No	Yes	Yes	SDN and NFV (Network Function Virtualization)	Yes	High	High	No	No	No
[22]	No	Yes	No	No	Yes	No	No	Yes	High	—	No	Yes	Yes
[23]	No	Yes	No	No	Yes	No	Cooja Simulator	Yes	High	High	No	Yes	Yes
[24]	No	Yes	Yes	No	Yes	Yes	Ethereum	Yes	High	High	Yes	Yes	Yes
[25]	No	Yes	No	No	No	No	No	Yes	High	—	Yes	Yes	Yes
[26]	No	Yes	Yes	No	Yes	Yes	Private BC	No	High	Low	Yes	Yes	Yes
[27]	No	Yes	Yes	No	Yes	No	Single Trusted Server	No	Low	High	Yes	Yes	Yes
[28]	No	Yes	Yes	No	Yes	No	Single Trusted Server	No	Low	High	Yes	No	No
our proposed work	Yes	Yes	Yes	Yes	Yes	Yes	Node.js	Yes	High	Low	No	No	No

The proposed mechanism is a decentralized framework for cross-domain authentication and authorization. It performs platform verification before user/IoT device authorization. The proposed framework is based on BC therefore, it has high fault tolerance. Moreover, it preserves user/IoT device privacy by allowing him to get authentication in his parent's domain. Similarly, the proposed framework is implementation on private BC therefore, its computation cost is low as compared to public BC. Moreover, it performs the cross-domain permission delegation. The proposed framework is secured against DoS, DDoS, Spoofing, and Sybil attacks.

VI. SECURITY ANALYSIS AND THREAD MODELS

In this section, we perform security analysis and discuss thread models of the proposed framework. Additionally, we compare the proposed framework with existing related frameworks in the literature.

A. SECURITY ANALYSIS

We analyzed the security of the proposed xDBAuth using Confidentiality, Integrity, Availability, Authentication/Authorization, and Non-repudiation (CIAAN) model. The CIAAN model security requirements are achieved by

TABLE 9. Evaluation of Security parameters.

Parameters	Description
Confidentiality	TLS is used to secure communication between user/IoT device, BC manager and BC.
Integrity	To ensures platform and data in transit integrity, SHA-256 hash function is implemented.
Availability	Achieved by using BC ledgers replication mechanism. Furthermore, the attacks on availability of the system likes DoS, DDoS are avoided by using platform verification.
Authentication/ Authorization	Proof-of-Authenticity/Integrity is used to authenticate users. Both internal and external users are authorized using delegation policies stored on BC.
Non-repudiation	Digital signature are used to achieve non-repudiation. Every user/IoT device must cryptographically signed his transaction.

integrating BC technology in xDBAuth architecture. The analysis of xDBAuth is summarized in **Table 9**.

1) CONFIDENTIALITY

It refers to hiding data from being accessed by illegal users. TLS is used to secure communication between user/IoT devices, BC manager, and BC.

2) INTEGRITY

It protects data from being changed illegally. In xDBAuth, both platform and data in transit are protected using BC cryptographic hash function i.e., SHA-256. By design, BC ledgers are immutable. In BC, each BC block in the chain contains the preceding block hash value.

3) AVAILABILITY

It ensures that the service or information is always available to legitimate users. The BC nodes have a copy of the distributed and timely synchronized ledger. Supposed an attacker damages the ledger at a certain node. The BC replication mechanism helps the node in restoring its original ledger from other peers. Furthermore, our proposed architecture authorizes users after the validation of the PoAI. PoAI ensures user authenticity and his platform integrity. Therefore, PoAI protects the network availability against different attacks likes DoS, DDoS, Spoofing, Sybil.

4) AUTHENTICATION AND AUTHORIZATION

Authentication is a mechanism to ensure that the user is genuine. The proposed authentication mechanism is based on PoAI. Also, PoAI ensures platform integrity. PoAI executes fast and requires low power. Similarly, authorization is a mechanism to ensure that the user is authorized for the given task. In xDBAuth, BC performs both internal and external user's authorization based on delegation policy validation. The delegation policies are stored in BC.

5) NON-REPUDIATION

It is a process to ensure that a user cannot deny his performed transactions. In the proposed architecture, every transaction is digitally signed with the user key. Hence, it provides non-repudiation to the user's transactions stored on the BC.

B. THREAT MODELS

In the proposed threat model, we define the following potential threats and their malicious behaviors.

1) MALICIOUS SR

Suppose a malicious SR wants to get authentication in the proposed framework by using a legitimate user ID. In the proposed framework every SR provides his platform hash and pseudonymous ID in the request. Before authorization, global BC verifies the user's platform hash from the user's parent domain. The parent domain stores user/IoT device pseudonymous IDs bounded with the user's platform hash. Therefore, it prevents the adversary from using other user's pseudonymous IDs to lunch a spoofing attack.

2) MALICIOUS SP

Suppose a malicious SP wants to know the real identity of the SR. In the proposed framework SRs provide their platform hashes and pseudonymous IDs to the SP for authentication and authorization. SR domain knows the real identity of the SR. Therefore, SP redirects SR to his parent domain. The SR domain provides the stored platform hash of the SR. It is very difficult for SP to extract SR real identity from his pseudonymous ID and platform hashes i.e., received in the request from the user or in the response from the user's parent domain.

3) MALICIOUS PARTNER DOMAIN

Suppose an adversary has compromised a peer of an IoT domain in the global BC network. The adversary wants to control the PoAI and disproves a legitimate user. The SP broadcasts user pseudonymous ID to all the peers i.e., domains for authentication. We assumed that each user belongs to a single domain. An SP may receive zero or more PoAI but only one will contain the actual hash. A malicious peer tries to find a user/IoT device platform hash using the provided pseudonymous ID of the user/IoT device. However, it is very difficult for the malicious peer the exact platform hash of the user/IoT device. But, it possible that the adversary may send a wrong platform hash in the PoAI. Therefore, in the proposed framework, the SP compares the hash received in the user request with the hash received in the PoAI. The hashes do not match and PoAI of the malicious peer is denied.

4) DENIAL-OF-SERVICE ATTACK (DoS) AND DISTRIBUTED DENIAL-OF-SERVICE (DDoS) ATTACK

In both DoS and DDoS attacks, the adversary overwhelms the selected node by using single or multiple compromised nodes. These attacks destroy the availability of the target device. Suppose an adversary has compromised a user/IoT device by installing malware. Now, the adversary wants to initiate a DoS/DDoS attack on other user/IoT devices by exploiting the delegation policy of the compromised user/IoT device. Therefore, the adversary sends a "T.access" transaction to the BC. However, our proposed framework performs user/IoT device authentication using PoAI. The compromised user/IoT device's current platform hash is different from his original hash received in the PoAI. Thus, the SP will not allow the compromised user/IoT device to access a resource.

VII. CONCLUSION

In this paper, we propose xDBAuth, a decentralized BC-based permission delegation and access control framework for IoT. The proposed framework provides access control for both internal and external users. Therefore, it consists of local and global smart contracts. The global smart contract allows user/IoT devices to get authentication in his parent IoT domain. Upon successful authentication, the global smart contract validates delegation policies stored on BC and makes an authorization decision.

The proposed PoAI mechanism uses platform hashes and provides strong privacy-preserving cross-domain authentication. Also, it replaces single trusted authorization service with BC, since it is a single point of failure and vulnerable to different attacks. Furthermore, we have implemented xDBAuth using Node.js. The results show that the cryptographic functions implemented in the proposed xDBAuth framework have negligible computational cost. Also, the proposed framework produces high throughput in an environment having a large number of concurrent requests. In the future, we will work on formal modeling and formal verification of the xDBAuth mechanism.

ACKNOWLEDGMENT

The authors would like to thank HEC Pakistan for research funding under the National Center for Cyber Security initiative for the project Provable Security of Blockchain Technologies.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[2] J. Rivera and R. van der Meulen, *Gartner Says 4.9 Billion Connected 'Things' Will Be in Use in 2015*. Stamford, CT, USA: Gartner, 2014.

[3] T. H. Payne, D. E. Detmer, J. C. Wyatt, and I. E. Buchan, "National-scale clinical information exchange in the united kingdom: Lessons for the united states," *J. Amer. Med. Inform. Assoc.*, vol. 18, no. 1, pp. 91–98, Jan. 2011.

[4] M. Alam, X. Zhang, K. Khan, and G. Ali, "XDAuth: A scalable and lightweight framework for cross domain access control and delegation," in *Proc. 16th ACM Symp. Access Control Models Technol. (SACMAT)*, 2011, pp. 31–40.

[5] Q. Alam, M. Alam, G. Ali, F. Azim, K. H. Khan, T. Ali, M. Ali, and A. Hayat, "Towards a formal framework for cross domain access control," *Int. Inf. Inst. (Tokyo). Inf.*, vol. 15, no. 10, p. 4303, 2012.

[6] H. Anada, J. Kawamoto, J. Weng, and K. Sakurai, "Identity-embedding method for decentralized public-key infrastructure," in *Proc. Int. Conf. Trusted Syst.* Beijing, China: Springer, 2014, pp. 1–14.

[7] S. Sheikh and A. K. Chaturvedi, "Analysis of sensitive data security on trusted third party in cloud computing," *Management*, vol. 17, p. 18, Apr. 2014.

[8] Q. Alam, S. Tabbasum, S. Malik, M. Alam, T. Tanveer, A. Akhunzada, S. Khan, A. Vasilakos, and R. Buyya, "Formal verification of the xDAuth protocol," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 9, pp. 1956–1969, Sep. 2016.

[9] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.

[10] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.

[11] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[12] V. Buterin, "A next-generation smart contract and decentralized application platform," Ethereum Found., White Paper 3 (37), 2014.

[13] V. Buterin, "On public and private blockchains," Ethereum, Ethereum Blog 7, Tech. Rep., Aug. 2015.

[14] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: Essential requirements and design options," 2016, *arXiv:1612.04496*. [Online]. Available: http://arxiv.org/abs/1612.04496

[15] (2014). *Ethereum Foundation. The Solidity Contract-Oriented Programming Language*. [Online]. Available: https://github:com/ethereum/solidity

[16] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf. (EuroSys)*. New York, NY, USA: ACM, 2018, p. 30.

[17] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Sep. 2018, pp. 264–276.

[18] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[19] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.* Neuchâtel, Switzerland: Springer, 2017, pp. 206–220.

[20] N. Tapas, G. Merlino, and F. Longo, "Blockchain-based IoT-cloud authorization and delegation," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2018, pp. 411–416.

[21] A. G. Abbasi and Z. Khan, "VeidBlock: Verifiable identity using blockchain and ledger in a software defined network," in *Proc. Companion 10th Int. Conf. Utility Cloud Comput. (UCC Companion)*, 2017, pp. 173–179.

[22] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and solutions," 2016, *arXiv:1608.05187*. [Online]. Available: http://arxiv.org/abs/1608.05187

[23] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 618–623.

[24] R. Xu, Y. Chen, E. Blasch, and G. Chen, "BlendCAC: A BLockchain-enabled decentralized capability-based access control for IoTs," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1027–1034.

[25] B. Tang, H. Kang, J. Fan, Q. Li, and R. Sandhu, "IoT passport: A blockchain-based trust framework for collaborative Internet-of-Things," in *Proc. 24th ACM Symp. Access Control Models Technol. (SACMAT)*. New York, NY, USA: ACM, 2019, pp. 83–92.

[26] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "FairAccess: A new blockchain-based access control framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5943–5964, Dec. 2016.

[27] S. Gusmeroli, S. Piccione, and D. Rotondi, "IoT access control issues: A capability based approach," in *Proc. 6th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, Jul. 2012, pp. 787–792.

[28] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An OAuth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors J.*, vol. 15, no. 2, pp. 1224–1234, Feb. 2015.

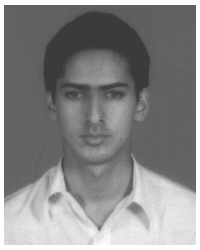
[29] H. Tan, G. Tsudik, and S. Jha, "MTRA: Multiple-tier remote attestation in IoT networks," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2017, pp. 1–9.

[30] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "Invited—Things, trouble, trust: On building trust in IoT systems," in *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, 2016, pp. 1–6.

GAUHAR ALI received the M.S. degree in computer science from the Institute of Management Sciences, Peshawar, in 2012. He is currently a Ph.D. Scholar with the University of Peshawar. His research interests include the Internet of Things, access control, blockchain, intelligent transport system, formal verification, and model checking.

NAVEED AHMAD received the B.S. degree in computer science from the University of Peshawar, Pakistan, in 2007, and the Ph.D. degree in computer science from the University of Surrey, U.K., in 2013. He is currently working as an Assistant Professor with the Department of Computer Science, University of Peshawar. His research interests include security and privacy in emerging networks, such as VANETs, DTN, and the Internet of Things (IoT).

YUE CAO (Member, IEEE) received the Ph.D. degree from the Institute for Communication Systems (ICS), University of Surrey, Guildford, U.K., in 2013. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, China. His research interest focuses on intelligent transport systems. He is also the Associate Editor of IEEE Access, *EURASIP Journal on Wireless Communications and Networking* (Springer), *KSII Transactions on Internet and Information Systems*, and the *International Journal of Vehicular Telematics and Infotainment Systems* (IGI Global).



SHAHZAD KHAN received the B.S. degree in computer science and the M.S. degree in computer science from the University of Peshawar, in 2015 and 2018, respectively. He is currently working as a Research Assistant with the National Center for Cyber Security, University of Peshawar, Pakistan. His research interests are blockchain technologies, the Internet of Things (IoT), and machine learning.

HAITHAM CRUICKSHANK (Member, IEEE) worked in ICS (formerly CCSR) since January 1996 on several European research projects in the ACTS, ESPRIT, Ten-Telecom, and IST programs. His main research interests are network security, satellite network architectures, VoIP, and IP conferencing over satellites. He has worked in several FP6 projects, such as SATLIFE, EuroNGI, and SATNEX also taught in the Data and Internet Networking and satellite communication courses at the University of Surrey. He is a Chartered Engineer and a Corporate Member of the IEEE in the U.K., also a member of the Satellite and Space Communications Committee of the IEEE ComSoc, and is active in the ETSI BSM (Broadband Satellite Multimedia) and the IETF MSEC groups.

EJAZ ALI QAZI received the M.S. degree in computer science from IBMS, Agricultural University Peshawar, Pakistan, in 2008. He is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science, University of Peshawar, Pakistan. He is also an Assistant Professor with the Department of Computer Science, University of Peshawar. His research interests are network security, intelligent transport system security and privacy, and its efficiency.

AZAZ ALI received the B.S. degree in computer science from the Institute of Management Sciences, Peshawar, in 2014. He is currently pursuing the M.S. degree in computer science with the School of Computer Science, China West Normal University. His research interests include access control, blockchain, intelligent transport systems, and the Internet of Things.