Wireless Personal Communications (2024) 136:1331–1352
https://doi.org/10.1007/s11277-024-11266-1

ACS-IoT: Smart Contract and Blockchain Assisted Framework for Access Control Systems in IoT Enterprise Environment

Aqsa Rashid[1] · Asif Masood[1] · Atta ur Rehman Khan[2]

Accepted: 20 January 2024 / Published online: 26 June 2024

Abstract
Centralized access control systems are unsuitable for IoT due to their resource-constrained, heterogeneous, and dynamic nature. Blockchain-assisted decentralized access control systems exist for IoT, but those approaches are tokenization-based. In some cases, IoT devices are not part of the blockchain network due to which they cannot interact with the access control system directly. Instead, they need a trusted admin, management hub, or a fog node for permissions verification and access to resources. This paper presents a smart contract and blockchain-assisted framework for the access control systems in the IoT enterprise environment, called ACS-IoT. In the proposed framework, resource-constrained IoT devices belong to the blockchain network. Therefore, these devices can directly access the permitted resources without any centrally administered authority and management hub's verification. We used smart contract and Ethereum blockchain for the new framework. Smart contract allows automated enforcement of access policies and serves as an autonomous agent running exactly as programmed. The proposed framework is validated through implementation of the proof of concept, and implemented prototype is deployed and tested on the Ethereum test network. The obtained results confirm that usage of blockchain and smart contract can be used as access management technology in the IoT enterprise environment.

Keywords Access control · Enterprise environment · Ethereum · Internet of Thing (IoT) · Smart contract

1 Introduction

Internet of Things (IoT) has potential applications in multiple domains [1]. Several IoT applications automate daily life processes, including smart healthcare, smart waste management, intelligent military wearables, battlefield, intelligent transportation, parking monitoring, traffic management, and multiple other applications [2–4]. IoT enables

✉ Atta ur Rehman Khan
dr@attaurrehman.com

1 Department of Information Security, National University of Sciences and Technology, Islamabad, Pakistan

2 College of Engineering and Information Technology, Ajman University, Ajman, UAE

interconnection of self-configurable and intelligent sensor-equipped entities, and facilitates them to effectively collaborate for secure communication and access control [4].

An enterprise network environment requires access control and communication betweek the connecting devices. Central authority-dependent access control systems fulfill the needs of the devices within the enterprise networks. In a traditional enterprise network environment, machines do not belong to several management groups during their lifespan and are homogeneous. The devices are not resource constrained and are usually managed by a single centralized manager at a time. However, the IoT devices are heterogeneous and belong to several management groups during their lifespan [5], and can be managed by various managers at a time. As compared to traditional enterprise network devices, IoT devices are resource-constrained devices [6] in terms of memory, computing power, and energy. Due to these restrictions, conventional schemes [7–11] are not suitable for IoT access control in an enterprise environment.

Blockchain technology can be one of the potential solutions to support access control services of IoT due to its decentralized nature and cryptographic security [12]. In the literature, many researchers have presented decentralized blockchain-enabled access control schemes. In [13–19], researchers present schemes that are based on tokenization. Similarly, in [20–32], the researchers present schemes that require trusted admin, management hub, or fog node for access permission verification from the blockchain. To the best of our knowledge, IoT access control for enterprise network environment has not received much attention.

This research presents a new framework for managing *IoT* devices in enterprise networks to facilitate decentralized access control systems. The proposed solution is *SC* and blockchain technology-based. *SC* is used to define the access control rules and policy for protected resources. Blockchain deploys *SC*, and it is cryptographically secure, flexible, and adaptable. The *ACS-IoT* framework provides the following features to the enterprise network environment:

- *ACS-IoT* is for the enterprise network environments to facilitate resource-constrained, heterogeneous, and dynamic IoT devices applicable in various other use cases.
- All operations of *ACS-IoT* are transparent and secure as the records immutable once stored on the blockchain. Temper proof logs of all the functions are managed programmatically in *SC*, which is useful in auditability.
- In the *ACS-IoT*, *IoT* devices can belong to several management groups during their lifespan and can be managed by various regulators at a time. For example, as an IoT device, a mobile can be part of multiple management systems; thus, it involves various mobile node regulators simultaneously.
- Proposed *ACS-IoT* is a privacy-preserving access control system for IoT devices as the location of *IoT* and access policy remain hidden.
- In *ACS-IoT*, access permissions are verified and validated independently from the blockchain without the intervention of any central authority, trusted admin, or management hub.

In particular, the research work contributes a new *SC* and blockchain-assisted access control architecture for *IoT*, called *ACS-IoT. ACS-IoT* differs from other contributions [8–11] in the way that it applies a specific design to avoid employing a trusted admin, management hub, or fog node to translate access requests of resources from constrained *IoT* devices into JSON-RPC. Rather the IoT nodes belong to the blockchain network. Hence *IoT* nodes can directly interact with the *ACS-IoT* and can access resources. As opposed to other solutions [13–19], the design of *ACS-IoT* does not require token-based authentication for access control. As the solution application scenario is a standalone enterprise environment, the design operates in a single, smart contract rather than multiple contracts [20–32]. The Etherscan verified proof of concept implementation of the *SC* is available publically on Etherscan under the OSL-3.0 license. The corresponding contract ABI, Bytecode, and JSON interface are publically available on GitHub (https://github.com/ARashidMCSNUST/PoW-ACS-IoT). All tested results and interaction, together with the implemented *SC*, are publically available on Etherscan on the contract address (0xbe0856F49c3f08D-Ba2B70B489653390e4F4aD6eb) for verification and validation:

The rest of the paper is organized as follows: Sect. 2 presents a new smart contract and blockchain-assisted access control systems for *IoT*. Section 3 offers the technical model of the *ACS-IoT.* Afterward, Sect. 4 elaborates ACS-IoT workflow for service provisioning together with the detail of the simulation environment and performance analysis. Section 5 presents threat modeling using STRIDE for the security analysis of the *ACS-IoT*. Section 6 compares the *ACS-IoT* with the related work is contributed. Section 7 covers the conclusions and future contributions of the research.

2 ACS-IoT's System Architecture

This section first presents the assumptions associated with *ACS-IoT* and then presents the overview of the *ACS-IoT's* system components.

2.1 Associated Assumptions

ACS-IoT has the following associated assumptions:

- Challenger cannot break the standard cryptographic and blockchain's primitives.
- Challenger cannot control the majority of the hashing power of the blockchain network, i.e., 51%.

2.2 Overview of ACS-IoT

The *ACS-IoT*'s architecture is a new decentralized *IoT* access control system where the blockchain is used to access policy creation and verification. All the entities will be part of blockchain technology, including *IoT* devices. The majority of *IoT* devices are unable to store blockchain information due to their constrained nature. Consequently, our architecture employed Light Ethereum Subprotocol (LES) for the *IoT* devices. Besides these, a smart contract is used to register nodes, create policy, and verify access policy. Figure 1 shows the architecture of our system. The proposed framework relies on the six essential components described below.

2.2.1 Component A: The Blockchain Network

The Blockchain Network used by the proposed framework is a public Ethereum blockchain [33] that deploys smart contracts. We utilize a public blockchain rather than a private one to make the mechanism open to any number of IoTs. In other words, depending on a

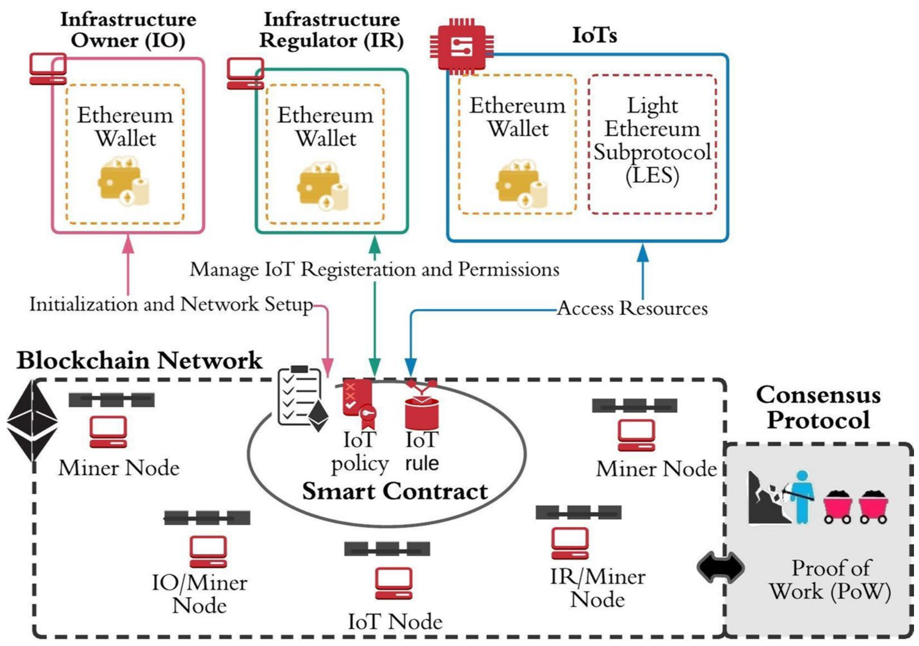


Fig. 1 SC and blockchain-assisted decentralized model of *ACS-IoT* for enterprise network environment

private blockchain makes *ACS-IoT* valid only for a predefined number of *IoT* devices. Once the system is installed, it will be impossible to add new *IoTs*, limiting the flexibility and the scalability of the approach considerably. All the interactions with *ACS-IoT* are considered a transaction validated by blockchain to be considered a valid operation.

2.2.2 Component B: The Consensus Protocol

The Consensus Protocol is an essential component of the proposed architecture as all the blocks added to the blockchain network require this protocol for the validity of the block. In *ACS-IoT*, the Proof of Work (*PoW*) consensus protocol is employed for block confirmation.

2.2.3 Component C: The Smart Contract

The Smart Contract works like an autonomous agent and contains the operations required to govern the proposed access control system *ACS-IoT*. The contract is used to update a distributed data store and promote trust in the system as the code is immutable after its deployment. The designed contract operations can be interacted in two modes: Views (*VW*) and transactions (*TX*).

TX is a call to a function that alters the contract's state, and this change must be stored on the blockchain. To do so, TX must be submitted to the miners. Therefore TX is also called the write contract function. Views are similar to the TX. The difference is that they do not change the contract's state at the blockchain level and thus do not need to store the blockchain's interaction. Any node that has a copy of the blockchain can execute views locally. As the VWs are not submitted to miners, this makes their use fast and free of cost.

VW is also known as reading contract functions. All contract interactions can be designed in two versions, first a full version where other contracts can access its information, and second a lighter version where it only emits events on blockchain. As the storage on Ethereum has associated expensive gas cost, the trade-off in the light version of the contract is that the light version only emits events on the blockchain. However, interaction with other contracts is not possible in a lighter version.

2.2.4 Component D: The Infrastructure Owner (*IO*)

The *IO* device is responsible for deploying the SC at the top of the blockchain network and register infrastructure regulators *(IR)*. It uses the *IO* Ethereum Wallet module to authenticate the *IO* devices and send the required Ether to the device to store its public key in the Ethereum blockchain. The IO owns the smart contract during the lifetime of the *ACS-IoT*. On accepting the smart-contract at blockchain, the smart contract is recognized by a unique address in the blockchain network. All smart-contracts' interacting nodes need to know the unique contract address to communicate with the contract to access resources.

2.2.5 Component E: Infrastructure Regulator (IR)

The *IR* node is the governing node responsible for registering *IoT* nodes and creating, updating, and revoking their access policies. It uses *the IR* Ethereum Wallet module to authenticate the *IR* devices, send the required Ether to the device to store its public key in the Ethereum blockchain, and perform governing operations.

2.2.6 Component F: Internet of Things (IoT)

The Internet of Things *(IoT)* is part of a blockchain network and can access resources from the *ACS-IoT* directly after policy verification by the blockchain. These are considered light clients, as the *IoT* nodes are resource-constrained devices and cannot store blockchain information. Due to their resource-constrained nature and limited computing power, these devices cannot participate in the mining process. In the proposed ACE-IoT, Light Ethereum Subprotocol (LES) is employed to handle light clients' storage and mining problems. The LES only downloads block headers as they emerge, and other blockchain components are fetched on request. They have complete access to the blockchain in a secure manner, but they do not mine and therefore do not engage in the consensus process.

3 Working of the Proposed Smart Contract and Blockchain Assisted *ACS-IoT* Framework

An alteration of record in the blockchain ledger is called a transaction. Transactions are collected by the *SC* service in the blocks and then verified in the blockchain (Fig. 2). The contract service has an application programming interface (API) to interact with *IO*, *IR*, and *IoTs* with the *ACS-IoT*. A contract engine translates *SC* into blocks of transactions for further verification by other network nodes. Key ledgers of the proposed network management framework are defined below:

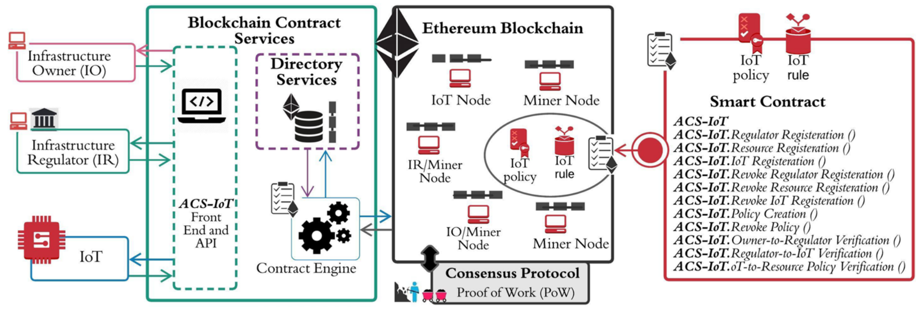

Infrastructure Owner (IO)
Infrastructure Regulator (IR)
IoT
Blockchain Contract Services
Directory Services
ACS-IoT Front End and API
Contract Engine
Ethereum Blockchain
IoT Node
Miner Node
IR/Miner Node
IoT policy
IoT rule
IO/Miner Node
Miner Node
Consensus Protocol
Proof of Work (PoW)
IoT policy
IoT rule
Smart Contract
ACS-IoT
ACS-IoT.Regulator Registeration ()
ACS-IoT.Resource Registeration ()
ACS-IoT.IoT Registeration ()
ACS-IoT.Revoke Regulator Registeration ()
ACS-IoT.Revoke Resource Registeration ()
ACS-IoT.Revoke IoT Registeration ()
ACS-IoT.Policy Creation ()
ACS-IoT.Revoke Policy ()
ACS-IoT.Owner-to-Regulator Verification ()
ACS-IoT.Regulator-to-IoT Verification ()
ACS-IoT.oT-to-Resource Policy Verification ()

Fig. 2 Technical model of *SC* and blockchain-assisted decentralized *ACS-IoT* for enterprise network environment

- *The R ledger* contains information about registered and revoked resources *R*, which are under IO's control.
- *The IR ledger* contains information about registered and revoked *IR*s by the *IO*.
- *The IoT ledger* contains information about registered and revoked IoTs under the control of specific IR.
- *The ledger* contains information about registered and revoked policies created for IoT-to-Resource permission verification by the *IR* of *IoT*.

We described the technical detail of operations defined in the *SC* of the *ACS-IoT* system, which consist of the following sets:

- $ID_{IR} = \{I_{IR_1}, I_{IR_2}, \ldots, I_{IR_n}\}$
- $ID_{IoT} = \{I_{IoT_1}, I_{IoT_n}, \ldots, I_{IoT_n}\}$
- $ID_R = \{I_{R_1}, I_{R_2}, \ldots, I_{R_n}\}$
- $P = p_1^{I_{IoT} \to I_R}, p_2^{I_{IoT} \to I_R}, p_3^{I_{IoT} \to I_R}, \ldots, p_n^{I_{IoT} \to I_R}$

where ID_{IR}: is the set of unique and unclonable identities of infrastructure regulator, ID_{IoT}: is the set of unique and unclonable identities of the *IoT* nodes, ID_R: is the set of unique and unclonable identities of the resources owned by *IO*, and P: is the policy set, where $p^{I_{IoT} \to I_R}$, states the authorizations that the *IoT* device with identity I_{IoT} has over the resource I_R owned by IO.

ACS-IOT's smart contract is designed to perform the following operations:

- *Regulator-Registration*

(I_{IO}, I_{IR}): Infrastructure owner IO uses this operation to register a unique infrastructure regulator I_{IR}, defined as:

$$ID_{IR}\prime \leftarrow ID_{IR} \bigcup \{I_{IR}\}$$

$$if I_{IR} \notin ID_{IR} and I_{IR} istheregulatingauthority$$

- *Resource-Registration*

(I_{IO}, I_R): This operation is designed to register unique resources by infrastructure owner I_{IO}, defined as:

$$ID_{R}\prime \leftarrow ID_R \bigcup \{I_R\}$$

$$if I_R \notin ID_R and I_{IO} is the infrastructure owner.$$

- *IoT-Registration*

(I_{IR}, I_{IoT}): Infrastructure regulator *IR* uses this operation to register a unique *IoT* device, defined as:
$ID_{IoT}\prime \leftarrow ID_{IoT} \bigcup \left\{I_{IoT}\right\}$ if $I_{IoT} \notin ID_{IoT}\ and I_{IR} is the regulating authority of I_{IoT}$

- *Revoke-Regulator-Registration*

(I_{IO}, I_{IR}): This operation is designed to revoke registeration of a registered regulator I_{IR} by infrastructure owner I_{IO}, defined as:

$$ID_{IR}\prime \leftarrow ID_{IR} \bigcap \{I_{IR}\}$$

$$if I_{IR} \in ID_{IR}, I_{IR} is the regulating authority$$

$$and Regulator - to - IoT verification = \emptyset$$

- *Revoke-Resource-Registration*

(I_{IO}, I_R): This operation is designed to revoke the registration of a registered resource I_R by infrastructure owner I_{IO}, defined as:

$$ID_{R}\prime \leftarrow ID_R \bigcap \{I_R\}$$

$$if I_R \in ID_R and I_{IO} is the infrastructure owner$$

$$and IoT - to - Resource verification = \emptyset$$

- *Revoke-IoT-Registration*

(I_{IR}, I_{IoT}): This operation is designed to revoke the registration of a registered I_{IoT} by infrastructure regulator I_{IR} of I_{IoT}, defined as:
$ID_{IoT}\prime \leftarrow ID_{IoT} \bigcap \left\{I_{IoT}\right\}$ if $I_{IoT} \in ID_{IoT}\ and I_{IR} is the regulating authority of I_{IoT}$

- *Policy-Creation*

$(I_{IR}, I_{IoT}, I_R, p)$: This operation is designed to create access policy by defining rule and permissions between IoT and R, defined as:

$P\prime \leftarrow P \bigcup \{p^{I_{IoT} \rightarrow I_R}\}$

$ifI_{IR} \in ID_{IR}, I_{IoT} \in ID_{IoT}, I_R \in ID_R, p^{I_{IoT} \rightarrow I_R} \notin P,$ if I_{IR} is the regulating authority of I_{IoT}

- *Revoke-policy*

$(I_{IR}, I_{IoT}, I_R, p)$: This operation is designed to revoke access policy, defined as:$P\prime \leftarrow P \bigcap \{p^{I_{IoT} \rightarrow I_R}\}$

$ifI_{IR} \in ID_{IR}, I_{IoT} \in ID_{IoT}, I_R \in ID_R, p^{I_{IoT} \rightarrow I_R} \in P,$ if I_{IR} is the regulating authority of I_{IoT}

- *Owner-to-Regulator Verification*

This operation returns the set of all *IRs* managed by the infrastructure owner *IO, defined as:*

$$\left\{I_{IR} | I_{IR} \exists I_{IO} : I_{IO} istheownerforI_{IR}\right\}, ifI_{IR} \in ID_{IR}$$

- *Regulator-to-IoT Verification*

This operation returns the set of all *IoTs* managed by specific regulator *IR,* defined as:

$$\left\{I_{IoT} | I_{IoT} \exists ID_{IR} : I_{IR} istheregulatingauthorityforI\right\} oT,$$

$$ifI_{IoT} \in ID_{IoT}$$

- *IoT-to-Resource Policy Verification*

This operation returns the set of all permissions of *IoT* over the resource R according to defined policy, defined as:

$$\left\{I_{IoT} | I_{IoT} \exists ID_{IoT} \wedge p^{I_{IoT} \rightarrow I_R} \in P\right\}$$

- The *IO*, *IR*, and *P* are recognized by the unique public and private key pair. In contrast, the identity of the *R* is recognized by auto-generated unique and unclonable id. Access permissions for the *IoT* to view, read, write, or execute the specific *R* can be defined using *the smart contract's Policy_Creation ()* operation. The following rule depicts the access decision criteria of the proposed *ACS-IoT* framework.

$$Access\_Decision(I_{IoT}, I_R, \{p^{I_{IoT} \rightarrow I_R}\}) = \begin{cases} Grant, if(I_{IoT} \in ID_{IoT}, I_R \in ID_R, \{p^{I_{IoT} \rightarrow I_R}\} \in P) \\ Deny, Otherwise \end{cases}$$

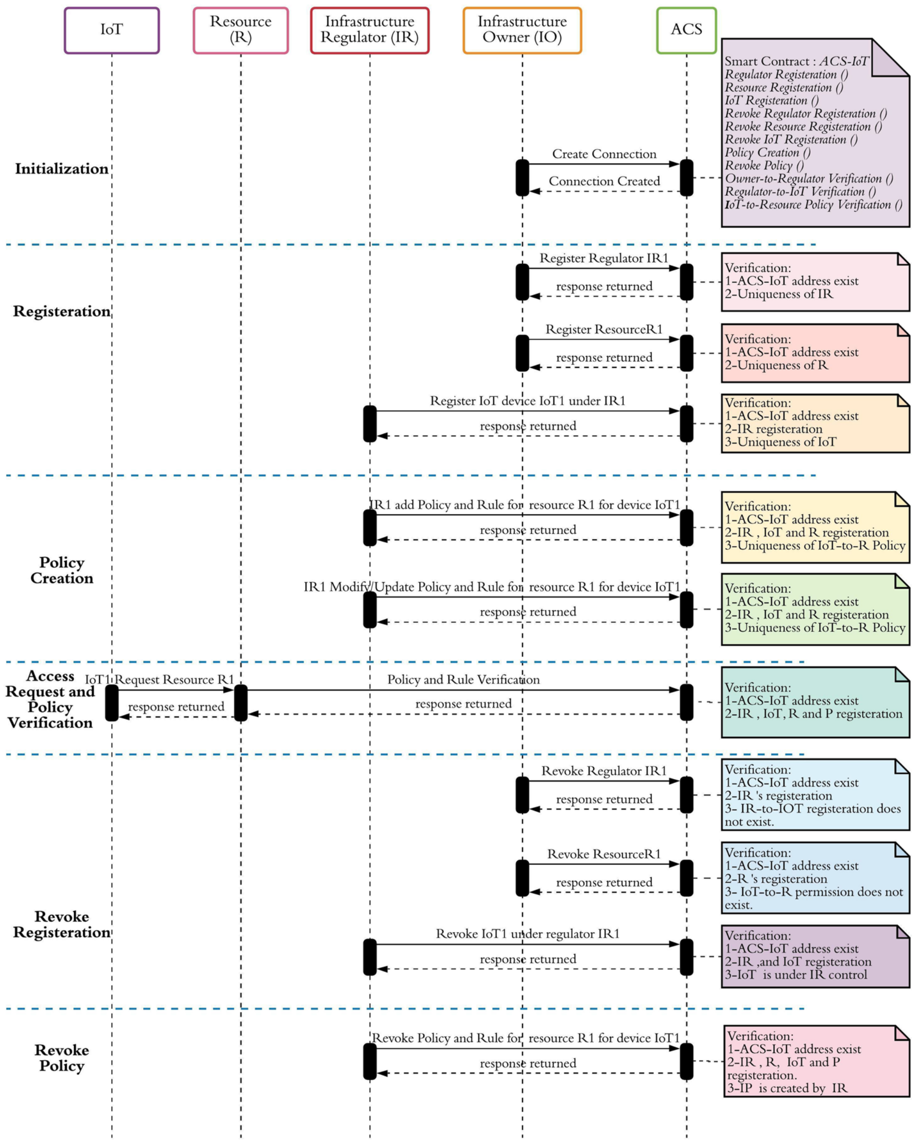

IoT
Resource (R)
Infrastructure Regulator (IR)
Infrastructure Owner (IO)
ACS
Smart Contract : ACS-IoT
Regulator Registeration ()
Resource Registeration ()
IoT Registeration ()
Revoke Regulator Registeration ()
Revoke Resource Registeration ()
Revoke IoT Registeration ()
Policy Creation ()
Revoke Policy ()
Owner-to-Regulator Verification ()
Regulator-to-IoT Verification ()
IoT-to-Resource Policy Verification ()
Initialization
Create Connection
Connection Created
Registeration
Register Regulator IR1
response returned
Verification: 1-ACS-IoT address exist 2-Uniqueness of IR
Register ResourceR1
response returned
Verification: 1-ACS-IoT address exist 2-Uniqueness of R
Register IoT device IoT1 under IR1
response returned
Verification: 1-ACS-IoT address exist 2-IR registeration 3-Uniqueness of IoT
Policy Creation
IR1 add Policy and Rule for resource R1 for device IoT1
response returned
Verification: 1-ACS-IoT address exist 2-IR , IoT and R registeration 3-Uniqueness of IoT-to-R Policy
IR1 Modify/Update Policy and Rule for resource R1 for device IoT1
response returned
Verification: 1-ACS-IoT address exist 2-IR , IoT and R registeration 3-Uniqueness of IoT-to-R Policy
Access Request and Policy Verification
IoT1 Request Resource R1
response returned
Policy and Rule Verification
response returned
Verification: 1-ACS-IoT address exist 2-IR , IoT, R and P registeration
Revoke Registeration
Revoke Regulator IR1
response returned
Verification: 1-ACS-IoT address exist 2-IR 's registeration 3- IR-to-IOT registeration does not exist.
Revoke ResourceR1
response returned
Verification: 1-ACS-IoT address exist 2-R 's registeration 3- IoT-to-R permission does not exist.
Revoke IoT1 under regulator IR1
response returned
Verification: 1-ACS-IoT address exist 2-IR ,and IoT registeration 3-IoT is under IR control
Revoke Policy
Revoke Policy and Rule for resource R1 for device IoT1
response returned
Verification: 1-ACS-IoT address exist 2-IR , R, IoT and P registeration. 3-IP is created by IR

Fig. 3 Protocol diagram of the *SC* and blockchain-assisted decentralized model of *ACS-IoT* for enterprise network environment

4 Proposed Workflow of *ACS-IoT* for Service Provisioning in Enterprise Network Environment

The section first explains the interaction with the proposed *ACS-IoT* through a protocol diagram and then presents the simulation environment and performance analysis in the proceeding sub-section.

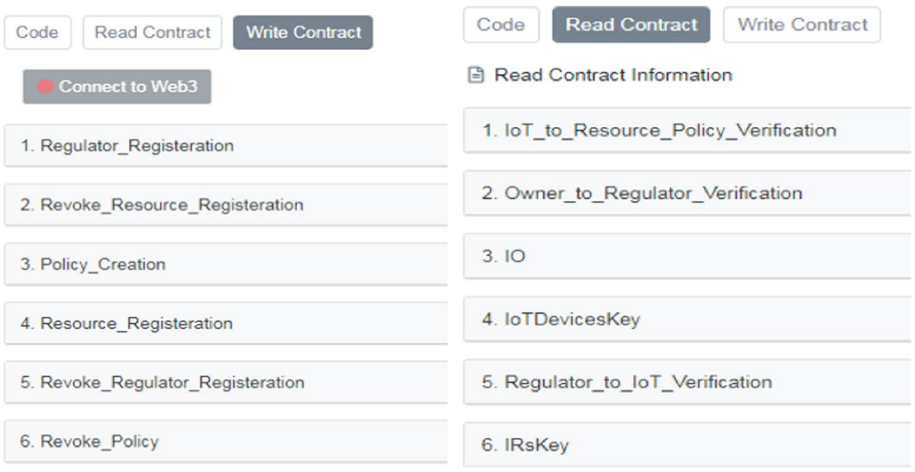
4.1 Generalized Workflow for Service Provisioning

Figure 3 depicts the blockchain-assisted smart contract protocol diagram between the *IO*, *IR*, *IoT*, *R*, and *ACS*. Different phases of interaction include initialization, registration, policy creation, access request, and policy verification, revoke registration and revoke policy, described below.

- *Initialization* Initially, the IO creates ACS in the blockchain network. The *IO* establishes a connection by deploying the smart contract into the top of the blockchain. Successful deployment creates a unique, unclonable address for the ACS-*IoT*, identified as the *ACS-IoT* identity. All interacting nodes identified the *ACS-IoT* by this address.
- *Registration* This phase includes the registration of unique R, IR, and their regulated *IoT*s. *IO* first registers all of available resources R at the blockchain level. Then it registers the IR and enables IR to govern the registration and access operation for *IoTs*. IR uses the *IoT* registration command and registers unique *IoT* nodes. Registered *IoT*s are under the control of corresponding IR.
- *Policy Creation* In the policy creation phase, *IR* defines an access policy for their registered *IoTs* and stores it at the blockchain to access and verify globally.
- *Access Request and Policy Verification* When an IoT node requests access to a resource, the verification operation verifies the blockchain's policy and returns the response in the form of a grant or deny depending upon verification.

Table 1 Implementation detail of the contract functions

TX: Write Contract Functions	VW: Read Contract Functions
TX1:Resource_Registeration Purpose: Register unique resource *R* owned by *IO* Invoke Rule: IO must sign TX1	VW1: IO Purpose: It returns the detail of the infrastructure owner *IO* Invoke Rule: Public
TX2: Regulator_Registeration Purpose: Register unique resource *R* owned by *IO* Invoke Rule: IO must sign TX2	VW2: Owner_to_Regulator_Verification Purpose: It returns the detail of *IRs* managed by the infrastructure owner *IO* Invoke Rule: Public
TX3: Policy_Creation Purpose: Register unique resource *R* owned by *IO* Invoke Rule: IR must sign TX3	VW3: Regulator_to_IoT_Verification Purpose: It returns the detail of *IoTs* managed by the infrastructure regulator *IR* Invoke Rule: Public
TX4: Revoke_Resource_Registeration Purpose: Revoke resource *R* registration owned by *IO* Invoke Rule: IO must sign TX4	VW4: IoT_to_Resource_Verification Purpose: It returns the detail of access permission of *IoTs* over to a resource Invoke Rule: Public
TX5: Revoke_Regulator_Registeration Purpose: Revoke register I*R* registration Invoke Rule: IO must sign TX5	VW5: IoTDevicesKey Purpose: It returns the index position of a specific IoT device stored on the contract Invoke Rule: Public
TX6: Revoke_Policy Purpose: Revoke the created policy for IoT by IR Invoke Rule: IR must sign TX6	VW6: IoTDevicesKe Purpose: It returns the index position of a specific IoT device stored on the contract Invoke Rule: Public

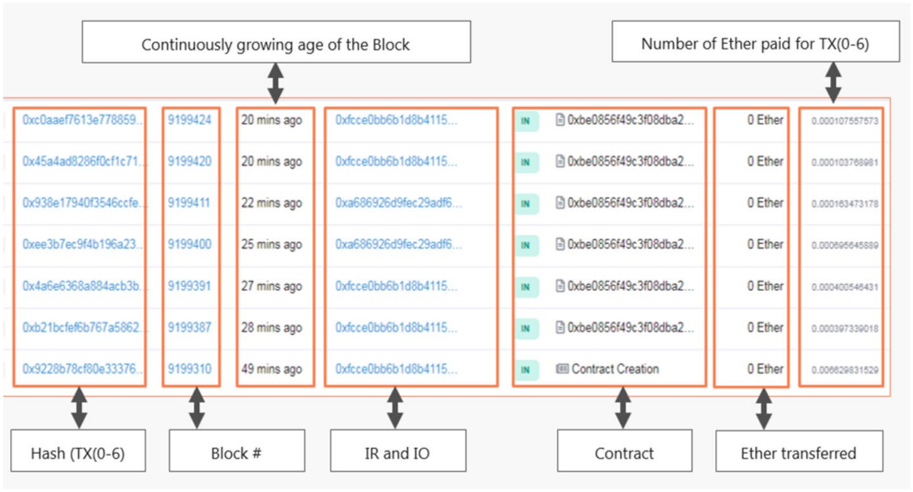


1. Regulator_Registeration
2. Revoke_Resource_Registeration
3. Policy_Creation
4. Resource_Registeration
5. Revoke_Regulator_Registeration
6. Revoke_Policy

1. IoT_to_Resource_Policy_Verification
2. Owner_to_Regulator_Verification
3. IO
4. IoTDevicesKey
5. Regulator_to_IoT_Verification
6. IRsKey

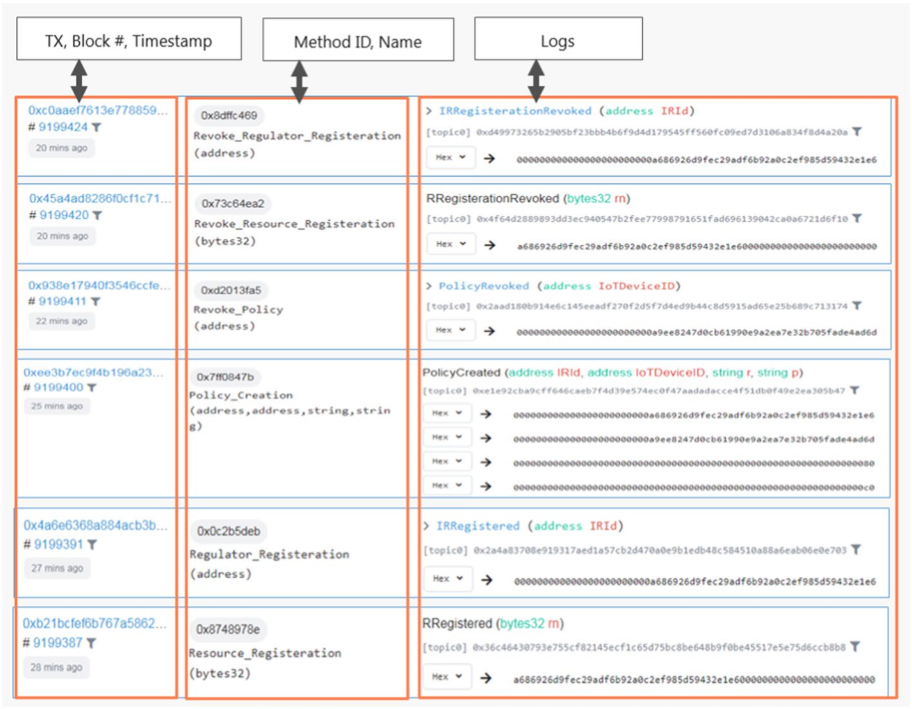
Fig. 4 *ACS-IoT* API for reading the contract and write a contract

- *Revoke Registration* Revoke registration includes the revocation of registration of *IR*, *R*, and *IoT*s. The *IO* can revoke the *IR* and *R* under his control, and *IR* can cancel the registration of *IoT*s under his control.
- *Revoke Policy* In this phase, *IR* revokes the created policy for *IoT*.


Continuously growing age of the Block
Number of Ether paid for TX(0-6)
Hash (TX(0-6)
Block #
IR and IO
Contract
Ether transferred

0xc0aaef7613e778859...	9199424	20 mins ago	0xfcce0bb6b1d8b4115...	IN	0xbe0856f49c3f08dba2...	0 Ether	0.000107557573
0x45a4ad8286f0cf1c71...	9199420	20 mins ago	0xfcce0bb6b1d8b4115...	IN	0xbe0856f49c3f08dba2...	0 Ether	0.000103768981
0x938e17940f3546ccfe...	9199411	22 mins ago	0xa686926d9fec29adf6...	IN	0xbe0856f49c3f08dba2...	0 Ether	0.000163473178
0xee3b7ec9f4b196a23...	9199400	25 mins ago	0xa686926d9fec29adf6...	IN	0xbe0856f49c3f08dba2...	0 Ether	0.000695645889
0x4a6e6368a884acb3b...	9199391	27 mins ago	0xfcce0bb6b1d8b4115...	IN	0xbe0856f49c3f08dba2...	0 Ether	0.000400546431
0xb21bcfef6b767a5862...	9199387	28 mins ago	0xfcce0bb6b1d8b4115...	IN	0xbe0856f49c3f08dba2...	0 Ether	0.000397339018
0x9228b78cf80e33376...	9199310	49 mins ago	0xfcce0bb6b1d8b4115...	IN	Contract Creation	0 Ether	0.006629831529

Fig. 5 Status of ACS-IoT's contract metadata after TX execution including H (TX), Block #, age of the block, from, to, value, and Ether



```
0xc0aaef7613e778859...   0x8dffc469                            > IRRegisterationRevoked (address IRId)
# 9199424                Revoke_Regulator_Registeration        [topic0] 0xd49973265b2905bf23bbb4b6f9d4d179545ff560fc09ed7d3106a834f8d4a20a
20 mins ago              (address)                             Hex → 000000000000000000000000a686926d9fec29adf6b92a0c2ef985d59432e1e6

0x45a4ad8286f0cf1c71...  0x73c64ea2                            RRegisterationRevoked (bytes32 rn)
# 9199420                Revoke_Resource_Registeration         [topic0] 0x4f64d2889893dd3ec940547b2fee77998791651fad696139042ca0a6721d6f10
20 mins ago              (bytes32)                             Hex → a686926d9fec29adf6b92a0c2ef985d59432e1e6000000000000000000000000

0x938e17940f3546ccfe...  0xd2013fa5                            > PolicyRevoked (address IoTDeviceID)
# 9199411                Revoke_Policy                         [topic0] 0x2aad180b914e6c145eeadf270f2d5f7d4ed9b44c8d5915ad65e25b689c713174
22 mins ago              (address)                             Hex → 000000000000000000000000a9ee8247d0cb61990e9a2ea7e32b705fade4ad6d

0xee3b7ec9f4b196a23...   0x7ff0847b                            PolicyCreated (address IRId, address IoTDeviceID, string r, string p)
# 9199400                Policy_Creation                       [topic0] 0xe1e92cba9cff646caeb7f4d39e574ec0f47aadadacce4f51db0f49e2ea305b47
25 mins ago              (address,address,string,strin         Hex → 000000000000000000000000a686926d9fec29adf6b92a0c2ef985d59432e1e6
                         g)                                    Hex → 000000000000000000000000a9ee8247d0cb61990e9a2ea7e32b705fade4ad6d
                                                               Hex → 0000000000000000000000000000000000000000000000000000000000000080
                                                               Hex → 00000000000000000000000000000000000000000000000000000000000000c0

0x4a6e6368a884acb3b...   0x0c2b5deb                            > IRRegistered (address IRId)
# 9199391                Regulator_Registeration               [topic0] 0x2a4a83708e919317aed1a57cb2d470a0e9b1edb48c584510a88a6eab06e0e703
27 mins ago              (address)                             Hex → 000000000000000000000000a686926d9fec29adf6b92a0c2ef985d59432e1e6

0xb21bcfef6b767a5862...  0x8748978e                            RRegistered (bytes32 rn)
# 9199387                Resource_Registeration                [topic0] 0x36c46430793e755cf82145ecf1c65d75bc8be648b9f0be45517e5e75d6ccb8b8
28 mins ago              (bytes32)                             Hex → a686926d9fec29adf6b92a0c2ef985d59432e1e6000000000000000000000000
```

Fig. 6 Immutable logs of event stored on ACS-IOT's contract after successful execution of TXs for auditability

4.2 Simulation and Performance Analysis

We implement the blockchain-based mechanism for ACS-IoT *using a smart contract running on the Ethereum platform. The smart contract includes the following functions (*Table 1*):*

We deploy the *ACS-IoT's* smart contract on the Ethereum test network. Successful deployment of contract on the Ethereum network generates the following unique, unclonable contract address for the *ACS-IoT*:

0xbe0856F49c3f08DBa2B70B489653390e4F4aD6eb.

ACS-IoT's contract was created during the following TXHash#:

0×9228b78cf80e333768849b99592199ade92c05353cb622d6e8b06d4dabb5f71b.

The smart contract's complete code is deployed, verified, and published on the Etherscan, the Ethereum test network, under the OSL-3.0 license. Upon source code verification, the ByteCode and ABI for *ACS-IoT*'s contract address are successfully generated by Etherscan. The ByteCode and ABI of the contract, together with the JSON interface, is published on GitHub.

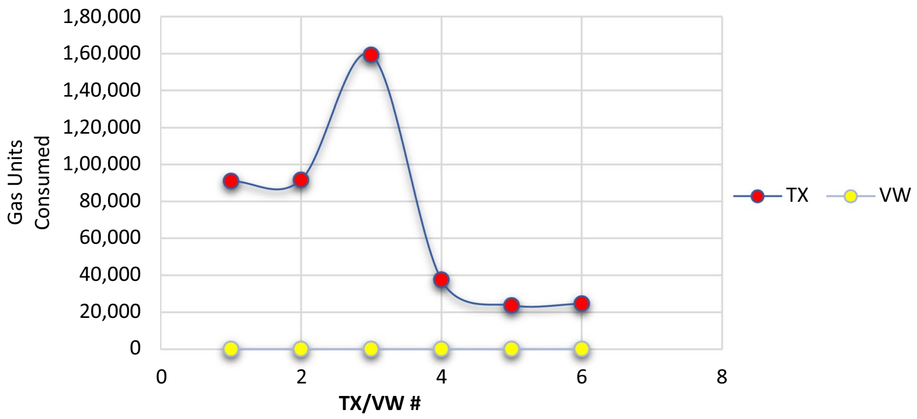


Fig. 7 Computation effort for various operations of *ACS-IoT*

Contract code and its interacting interface (Fig. 4) is publically available for verification, validation, and testing by anyone at the *ACS-IoT*'s contract address. To interact with the *ACS-IoT*'s contract, write a contract operation requires a connection to Web3. **Web3**.js allows interacting with a local or remote Ethereum node. However, interaction with the read contract operation is available publically without any connecting to Web3.js.

The uniquely generated contract is used as a synchronization system and decentralized database for the proposed methods. Figure 5 depicts *ACS-IoT's* synchronization system status and decentralized database after executing the contract's *TXs*. Figure 5 depicts the contract creation function's detail, including transaction attributes and block attributes as defined in [33]. For *TX1-TX6*, all these details are available on the *ACS-IoT's* contract on the Etherscan. Figure 6 shows the immutable logs of events stored on *ACS-IOT's* contract after successfully executing *TXs* for auditability.

4.3 Computational Effort

To measure the computational efforts for the implemented prototype's operation, we estimate the Ethereum gas unit consumption for all operations. Ethereum gas measures the amount of computational effort that it will take to execute certain functions. Graph of Fig. 8 shows the evaluation result of the computational effort of the *TXs* and *VWs* of the implemented prototype for *ACS-IoT* in terms of gas units consumed to execute *TXs* and *VWs*. Graphs of Fig. 7 show that the policy creation operation, *TX-3,* requires a higher computation effort than the other *TXs* of the contract. This operation requires many attributes that need more gas consumption for storage than the gas consumption of other *TXs*. VW operations do not consume any gas unit, as depicted in Fig. 7, because *VWs* operations are only read contract operations.

Various tests have been performed on the policy creation operation, *TX3*, with the variable payload. It has been observed that computational power increases with the increase in policy size. Graph of Fig. 8 depicts the variation in computational effort with different loads.

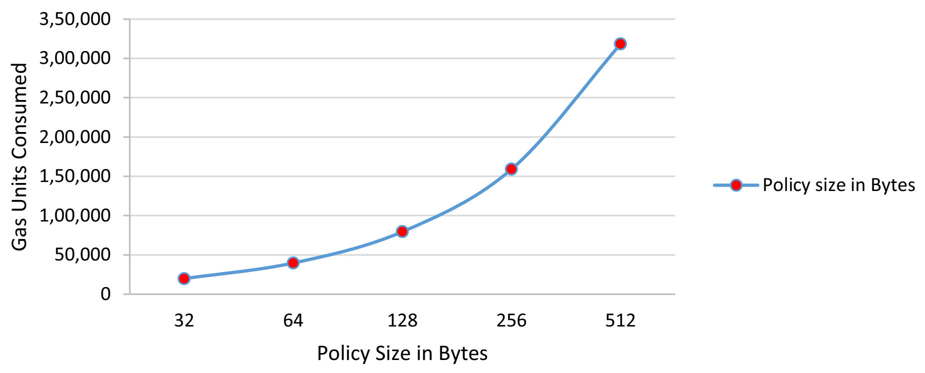


Fig. 8 Computation effort for variable policy (TX3) payload

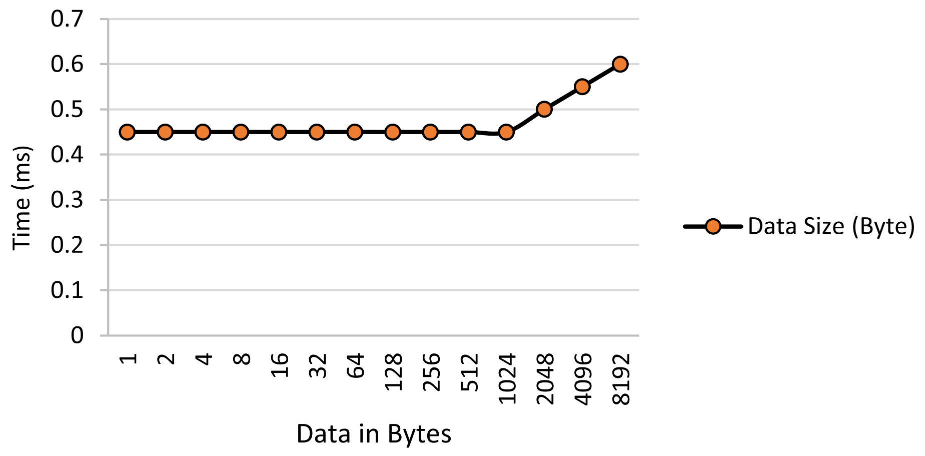


Fig. 9 Hashing and signature computation latency

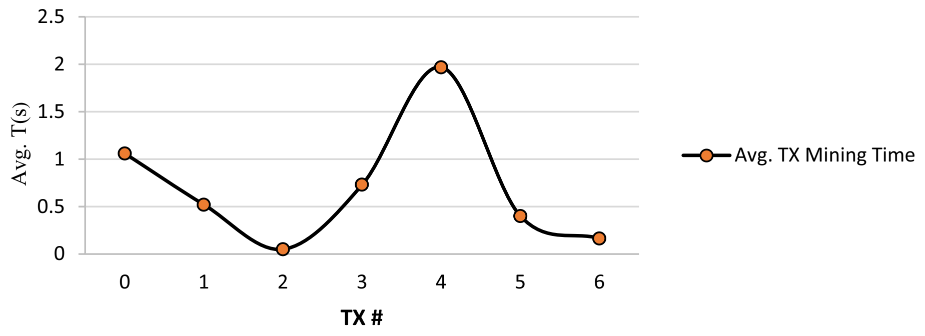


Fig. 10 Computation effort for various operations of *ACS-IoT*

4.4 Latency Analysis

Latency analysis has been divided into two following two categories:

- Hashing and Signature Computation Latency
- Mining Latency

4.4.1 Signature Computation Latency Analysis

When signing a transaction (TX), the transaction body is hashed with SHA3-256, and the output hash is signed with the ECDSA private key. Various experiments were carried out for various TX sizes. Since the output hash is a fixed 256 bytes, the hashing algorithm does more work with larger transaction sizes, while the signature computation latency remains the same.

Graph of Fig. 9 shows that for a payload size of 1024 bytes, the latency of signing a transaction remains constant as the latency of hashing with a data size of up to 1024 bytes is negligible. For the payload with a data size greater than 1024, the latency of hashing grows linearly while the latency of signature computation remains constant.

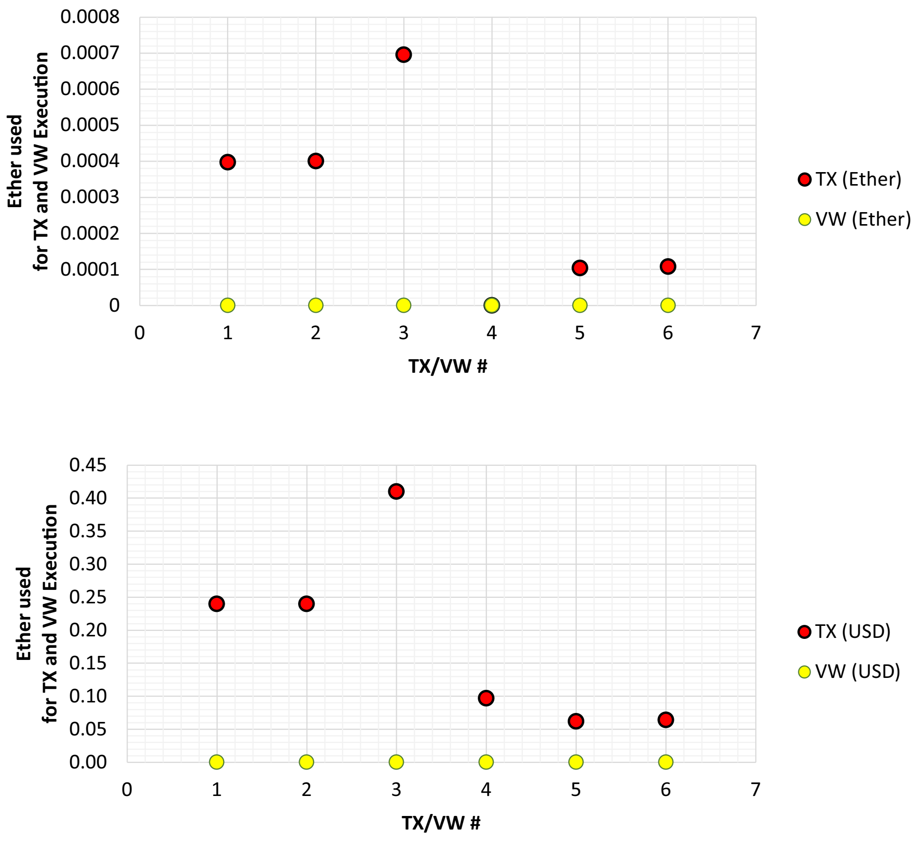


Fig. 11 Estimated Transaction fee, in Ether and USD, for Various Operations of *ACS-IoT*

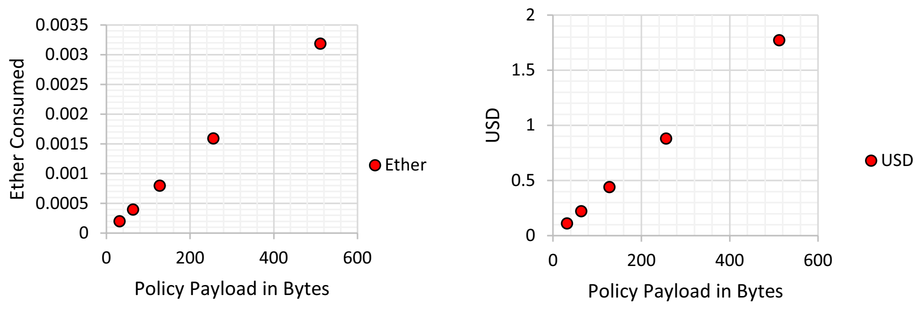


Fig. 12 Estimated Transaction fee, in Ether and USD, for variable policy (TX3) payload

4.4.2 Mining Latency Analysis

The mining latency is the time taken to mine a block and add it to the blockchain ledger. Graph of Fig. 10 shows the average mining time per transaction in a block containing a tested prototype *(TX-0 to TX-6).*

4.5 Economic Feasibility Analysis

To measure the approximate costs of running various ACS-IoT operations, we ran the functions of our implemented prototype in the Ethereum test network. We measured the approximate price in Ether and USD for creating the *ACS-IoT* at the blockchain and for each operation supported by the *ACS-IoT*. We tested the interaction with transactions on 05 Dec and computed the USD cost on 07 Dec, 1:44 pm UTC, when 1 Ether equals 596.04 USD. We paid the transaction fee at the rate of 0**.**000000004369772225 Ether (4**.**369772225 Gwei) per gas, which is the standard gas price at the 05 Dec 2020.

Figure 11 shows the evaluation result of cost estimation for the *TXs* of the implemented prototype in Ether and the USD. Graphs of Fig. 11 show that the policy creation operation, *TX-3,* requires a higher price in Ether and USD for its execution than the other *TXs* of the contract. T*X-3* requires a maximum number of attributes compared to that of other *TXs*. VW operations do not require any fee, as depicted in Fig. 11, because *VWs* operations are only read contract operations. Read contract does not change the contract's state; that is why these operations are free of cost.

Various tests have been performed on the policy creation operation, *TX3*, with the variable payload. It has been observed that cost increases with the increase in policy size. Figure 12 depicts the variation in cost in Ether and USD with different payloads.

The results show that barring fluctuations in Ether's gas price, or USD price, it is financially and technically feasible to deploy the *ACS-IoT system* on the Ethereum blockchain.

5 Threat Modelling Using STRIDE for Security Analysis of *ACS-IoT*

This section presents the threat modeling using STRIDE [34] (S of STRIDE: Spoofing, T of STRIDE: Tempering, R of STRIDE: Repudiation, I of STRIDE: Information Disclosure, D of STRIDE: Denial of Service, E of STRIDE: Elevation of Privileges) for security analysis of *ACS-IoT*. Following, we present the mapping between STRIDE threats and the security properties that guard against them in the *ACS-IoT*:

- *Spoofing→Authentication/Identification* An attacker can imitate *IR* and *IoT* devices' identities to make use of their privileges. Each *IR* and *IoT* device owns an unclonable and unique identity in the *ACS-IoT*. Only the legitimate *IR* and *IoT* device's identities are registered in the *ACS-IoT*. Consequently, the system can quickly identify registered devices and prevent spoofing.
- *Tempering→Integrity* An unauthorized *IR* can register *IoT* devices, and unauthorized *IoT* devices can access the enterprise network's protected resources. Moreover, an attacker can modify or alter registration or policy creation transaction data during transit over the web. Since all messages are signed with a private key of the corresponding owner node, the attacker requires the private key of a legitimate node to alter or substitute messages. However, only trusted nodes have been given the valid key-pairs at the registration and policy creation phase. All communications are considered *TX*, and each *TX* has a timestamp and needs a consensus phase to be correct. Thus, attackers' attempt reply messages will be rejected by the consensus mechanism.
- *Repudiation→Non-Repudiation* In *ACS-IoT-EB's* case, *IR* can deny the creation of policy for *IoT* devices. All the *TX*s are signed using the initiator's private key, known only by the initiator. Thus, it cannot deny the fact of signing a *TX*.
- *Information Disclosure→Confidential IoT* device location and access policy can create privacy concerns. The location of *IoT* and access policy remain hidden in *ACS-IoT*.
- *Denial of Services→Availability* a Denial of Service (DoS) or a Distributed DoS (DDoS) attack is an attacker's attempt to stop the authentic use of a provision. A DDoS attack against the identification registration mechanism can cause damages, including paralyzing the whole system or allowing unauthorized nodes to use the system. The decentralized architecture of blockchains makes the *ACS-IoT* robust against DoS/DDoS attacks and makes the system available 24/7. Transactions in Ethereum are costly, discouraging an attacker from spending money by sending many transactions for DoS/DDoS attacks.
- *Elevation of Privileges→Authorization* Elevation of privileges is when a user obtains a higher level of privilege than authorized. A malicious *IR* or *IoT* device may use the more elevated rights to compromise or destroy a system or unauthorized access information. Authorizations are defined using policy creation *TX* to prevent the elevation of privileges. *TX*. As a result, only the authorized *IR* and *IoT* devices can have access to resources and operations.

Table 2 Comparison of *ACS-IoT* with existing techniques

Category of Existing Techniques	Existing Techniques RELATED WORK	*ACS-IoT*
Centralized Access Control for IoT	• DAC, MAC, and RBAC [7] are central authority dependent, which is the bottleneck of a single point of failure and related security concerns of centralization. Moreover, these are not suitable for constrained environments and large-scale systems	• *ACS-IoT* differs in that it is a decentralized blockchain-based solution ideal for a constrained environment and large-scale systems
Decentralized Access Control System for IoT	• SmartOrBAC and CapBAC [8–11] are decentralized in nature. However, they do not support lightweight schemes	ACS-IoT differs in that it is suitable for lightweight clients, i.e., IoT devices, which belong to the blockchain network
Token and Smart contract-based Access Control for IoT using blockchain	• Token-based schemes [13–19] store access privileges in a digitally signed token, and access to resources require token verification. Smart contract-based methods [20–32] define and enforce access policies through a smart contract. In both cases, IoT devices do not belong to the blockchain network; preferably, these methods need a managing hub, trusted admin, or fog node for access permission assignment, verification, and validation. It makes these schemes bottleneck for a single point of failure	• *ACS-IoT* differs in that the method of *ACS-IoT* is not based on the protocols used by these schemes. The proposed system defines the IoT devices as the blockchain network node and permissions assignment and access verification, and validation is without the managing hub, trusted admin, fog node, or token verification

6 Comparison with the Existing Techniques

In [35], the researchers presented an extensive review of different access control mechanisms in *IoT*. The review classifies the existing access control solutions used in *IoT* and comprehensively discussed the commonly used Internet protocols applied in constrained environments. In this section, we compare the *ACS-IoT* with the schemes comprehensively reviewed in [35]. We did not discuss the detail of the architecture proposed by different researchers here. We only discuss the presented schemes' *IoT* limitations and compare how the proposed method provides a solution. We categorize the comparison (See Table 2) into the following three techniques for *IoT*:

7 Conclusion and Future Work

In an *IoT* enterprise environment, centralized access control systems are inefficient due to the heterogeneous, resource-constrained, and dynamic nature of IoTs. This research presented a smart contract and blockchain-based access control system in *an IoT* enterprise environment called *ACS-IoT*. Using *ACS-IoT,* organizations can manage the authentication and access to resources in a decentralized manner and eliminate centralization problems, such as single point of failure, limited transparency, security threats, privacy, and scalability concerns. *ACS-IoT* provides a generic, transparent, privacy-preserving, scalable, and easy-to-manage system for *IoT* devices. The implemented proof of concept prototype proves our design's efficiency, and the experimental results showed that *ACS-IoT* meets the required security requirements and is resilient against attacks. In the future, we plan to evolve the blockchain-based enforcement mechanism for inter-domain enterprise environments and optimize the implemented mechanism by defining Boolean functions. Moreover, the design and test of *ACS-IoT* with optimized miners, by employing Proof of Authority (PoA) and Proof of Stake (PoS) consensus protocol, is another consideration in future contribution.

Data Availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of Interest The authors have no relevant financial or non-financial interests to disclose.

References

1. By, G. S. (2016). More than half of major new business processes and systems will incorporate some element of the Internet of Things. *Publicado em Janeiro*.
2. Khan, M. A., & Khaled, S. (2018). IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems, 82*, 395–411.
3. Lee, I., & Kyoochun, L. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons, 58*(4), 431–440.
4. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*(7), 1645–1660.

5. Sun, X., & Nirwan, A. (2016). EdgeIoT: Mobile edge computing for the Internet of Things. *IEEE Communications Magazine, 54*(12), 22–29.
6. Sun, X., & Nirwan, A. (2017). Dynamic resource caching in the IoT application layer for smart cities. *IEEE Internet of Things Journal, 5*(2), 606–613.
7. UbaleSwapnaja, A., ModaniDattatray, G., & ApteSulabha, S. (2014). Analysis of dac mac rbac access control based models for security. *International Journal of Computer Applications, 104*(5), 6–13.
8. Ouaddah, A. (2015). Security analysis and proposal of new access control model on the Internet of Thing, In: *International conference on electrical and information technologies (ICEIT)*, IEEE.
9. Bouij-Pasquier, I., Abou E. I. K., A., Ouahman, A. A., & De Montfort, M. (2015). A security framework for internet of things. In: *International conference on cryptology and network security,* Springer, Cham, pp. 19–31.
10. Hernández, R., & José, L. (2013). Distributed capability-based access control for the Internet of Things. *Journal of Internet Services and Information Security (JISIS), 3*(3/4), 1–16.
11. Gusmeroli, S., Salvatore, P., & Domenico, R. (2013). A capability-based security approach to manage access control in the Internet of Things. *Mathematical and Computer Modelling, 58*(5–6), 1189–1205.
12. Nakamoto, S., Bitcoin, A. (2008). A peer-to-peer electronic cash system. Bitcoin. https://bitcoinorg/bitcoin. Pdf 4.
13. Ouaddah, A., Anas, A. E., & Abdellah, A. O. (2016). FairAccess: A new blockchain-based access control framework for the Internet of Things. *Security and Communications, 9*(18), 5943–5964.
14. Outchakoucht, A., Hamza, E. S., & Leroy, J. P. (2017). Dynamic access control policy based on blockchain and machine learning for the Internet of Things. *International Journal of Advanced Computer Science and Applications, 8*(7), 417–424.
15 Maesa, D. D. F., Paolo, M., & Laura, R. (2017). *Blockchain-based access control*. Springer, Cham: IFIP international conference on distributed applications and interoperable systems.
16. Alphand, O., Amoretti, M., Claeys, T., Dall'Asta, S., Duda, A., Ferrari, G., and Zanichelli, F. (2018). IoTChain: A blockchain security architecture for the Internet of Things. In: *2018 IEEE wireless communications and networking conference (WCNC),* 1–6, IEEE.
17. Xue, J., Xu, C., & Zhang, Y. (2018). Private blockchain-based secure access control for smart home systems. *KSII Transactions on Internet and Information Systems (TIIS), 12*(12), 6057–6078.
18. Xu, R., Chen, Y., Blasch, E., & Chen, G. (2018). Blendcac: A smart contract enabled decentralized capability-based access control mechanism for the IoT. *Computers, 7*(3), 39.
19. Fotiou, N., Pittaras, I., Siris, V. A., Voulgaris, S., & Polyzos, G. C. (2019). Secure IoT access at scale using blockchains and smart contracts. In: *2019 IEEE 20th international symposium on, a world of wireless, mobile and multimedia networks (WoWMoM)* IEEE, pp. 1–6.
20. Dorri, A., Kanhere, S. S., & Jurdak, R. (2016). Blockchain in the internet of things: challenges and solutions. *arXiv preprint* arXiv:1608.05187
21. Dorri, A., Kanhere, S. S., Jurdak, R., & Gauravaram, P. (2017). Blockchain for IoT security and privacy: The case study of a smart home. In: *2017 IEEE international conference on pervasive computing and communications workshops* (*PerCom workshops*), pp. 618–623 IEEE.
22. Huang, Z., Su, X., Zhang, Y., Shi, C., Zhang, H., & Xie, L. (2017). A decentralized solution for IoT data trusted exchange based-on blockchain. In: *2017 3rd IEEE international conference on computer and communications (ICCC),* pp. 1180–1184
23. Huh, S., Cho, S., & Kim, S. (2017). Managing IoT devices using a blockchain platform. In: *2017 19th international conference on advanced communication technology (ICACT)*, pp 464–467
24. Ali, M. S., Dolci, K., & Antonelli, F. (2017). IoT data privacy via blockchains and IPFS. In: *Proceedings of the seventh international conference on the internet of things*, pp. 1–7
25. Jo, B. W., Khan, R. M. A., & Lee, Y. S. (2018). Hybrid blockchain and internet-of-things network for underground structure health monitoring. *Sensors, 18*(12), 4268.
26. Novo, O. (2018). Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet of Things Journal, 5*(2), 1184–1195.
27. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., & Wan, J. (2018). Smart contract-based access control for the Internet of Things. *IEEE Internet of Things Journal, 6*(2), 1594–1605.
28. Jiang, Y., Wang, C., Wang, Y., & Gao, L. (2019). A cross-chain solution to integrating multiple blockchains for IoT data management. *Sensors, 19*(9), 2042.
29. Dorri, A., Kanhere, S. S., Jurdak, R., & Gauravaram, P. (2019). LSB: A lightweight scalable blockchain for IoT security and anonymity. *Journal of Parallel and Distributed Computing, 134*, 180–197.
30. Mbarek, B., Ge, M., & Pitner, T. (2020). Blockchain-based access control for IoT in smart home systems. In: *International Conference on Database and Expert Systems Applications* (pp. 17–32). Springer, Cham

31 Liu, H., Han, D., & Li, D. (2020). Fabric-IoT: A blockchain-based access control system in IoT. *IEEE Access, 8*, 18207–18218.
32. Bera, B., Saha, S., Das, A. K., & Vasilakos, A. V. (2020). Designing blockchain-based access control protocol in IoT-enabled smart-grid system. *IEEE Internet of Things Journal, 8*(7), 5744–5761.
33. Wood, G. (2014). Ethereum: A secure decentralized generalized transaction ledger. *Ethereum Project Yellow Paper, 151*(2014), 1–32.
34. Hernan, S., Lambert, S., Ostwald, T., & Shostack, A. (2006). Threat modeling-uncover security design flaws using the stride approach. *MSDN Magazine-Louisville*, pp. 68–75
35 Abdi, A. I., Eassa, F. E., Jambi, K., Almarhabi, K., & AL-Ghamdi, A. S. A. (2020). Blockchain platforms and access control classification for IoT systems. *Symmetry, 12*(10), 1663.

Aqsa Rashid received the M.C.S. degree (Hons.) in computer science and the M.S. degree in computer science with a specialization in information security from IUB, BWP, in 2013 and 2015, respectively. Ms. Aqsa is a Ph.D. research scholar at the Department of Information Security, Military College of Signals, National University of Sciences and Technology, (MCS, NUST), Pakistan. Aqsa Rashid earned the IBM Blockchain Badge from the IBM Blockchain Platform and Certified QAP Certification, from NUST, in 2018. She is actively involved in various research projects mainly related to the digitization of various industries through immutable blockchain technology. Her area of research interest includes cybersecurity, IoT, autonomous security services of organizations, enterprises, and military systems. The most important research contribution includes developing trustful systems for C2I, C3I, C4I, C5I, C6I, and C6ISR military systems, healthcare, Public Key Infrastructure (PKI), Privilege Management Infrastructure (PMI), and autonomous security services for organizational and enterprise Network systems. Ms. Aqsa Rashid has extensive experience in teaching, research, and administrative tasks. She has over 8 years of experience working in a multicultural dynamic environment at three universities in different cities in Pakistan. She has published several research papers in leading journals and conferences from IEEE, Elsevier, Springer, etc. She is serving a reviewer in most prestigious and reputed journals of IEEE, Springer, Elsevier, ACM, MDPI, Hindawi, Wiley, IET, and Taylor's and Francis since 2018.

Asif Masood is currently working as Professor and Dean of MCS, National University of Science and Technology. He has been awarded Research Productivity Award 2010-2011 by Pakistan Council for Science and Technology, Best Research Paper Award 2011 by Higher Education Commission (HEC) of Pakistan and Dr. M. N. Azam Prize in 2009 by Pakistan Academy of Sciences. He won First Prize in International Research Competition held in ICDAR-2011 in Beijing, China and won a Ph.D. scholarship, from Higher Education Commission (HEC), after an open competition all over Pakistan. He is a memberof the Editorial Board, Quarterly Journal of Science Technology and Development, Pakistan Council for Science and Technology. His research interest includes Information security, Cryptography, Network, and computer security. Currently, he is working on and supervising projects of Blockchain Technology.

Atta ur Rehman is an Associate Professor at College of Engineering and Information Technology, Ajman University, UAE. In the past, he has served as Postgraduate Program Coordinator at Sohar University, Director of National Cybercrime Forensics Lab Pakistan, and Head of Air University Cybersecurity Center. He has served as an Associate Editor of IEEE Access, Elsevier Journal of Network and Computer Applications, Associate Technical Editor of IEEE Communications Magazine, Editor of Springer Journal of Cluster Computing, Oxford Computer Journal, IEEE SDN Newsletter, KSII Transactions on Internet and Information Systems, and Ad hoc Sensor Wireless Networks journal. Moreover, he is a Senior Member of IEEE and ACM, and Steering Committee Member/ Track Chair/ Technical Program Committee (TPC) member of over 90 international conferences. He also serves as a domain expert for multiple international research funding bodies, and has received multiple awards, fellowships, and research grants. His areas of research interest include cybersecurity, mobile cloud computing, ad hoc networks, and IoT. For more updated information, visit his website at https://www.attaurrehman.com