




Article

Smart Contract-Based Access Control Framework for Internet of Things Devices

Md. Rahat Hasan ¹, Ammar Alazab ^{2,*} , Siddhartha Barman Joy ¹, Mohammed Nasir Uddin ^{1,*},
Md Ashraf Uddin ^{3,*} , Ansam Khraisat ³, Iqbal Gondal ⁴, Wahida Ferdose Urmi ¹ and Md. Alamin Talukder ¹ 

- ¹ Department of Computer Science and Engineering, Jagannath University, Dhaka 1100, Bangladesh; rahatcsejnu@gmail.com (M.R.H.); siddharthjoy88@gmail.com (S.B.J.); wahidaferdoseurmi@gmail.com (W.F.U.); alaminatalukder.cse.jnu@gmail.com (M.A.T.)
² Centre for Artificial Intelligence and Optimization, Torrens University, Brisbane, QLD 4006, Australia
³ School of Information Technology, Deakin University, Warrnambool Campus, Geelong, VIC 3125, Australia; ansam.khraisat@deakin.edu.au
⁴ School of Computing Technologies, STEM College, RMIT University, Melbourne, VIC 3001, Australia; iqbal.gondal@rmit.edu.au
* Correspondence: ammar.alazab@torrens.edu.au (A.A.); nasir@cse.jnu.ac.bd (M.N.U.); ashraf.uddin@deakin.edu.au (M.A.U.)

Abstract: The Internet of Things (IoT) has recently attracted much interest from researchers due to its diverse IoT applications. However, IoT systems encounter additional security and privacy threats. Developing an efficient IoT system is challenging because of its sophisticated network topology. Effective access control is required to ensure user privacy in the Internet of Things. Traditional access control methods are inappropriate for IoT systems because most conventional access control approaches are designed for centralized systems. This paper proposes a decentralized access control framework based on smart contracts with three parts: initialization, an access control protocol, and an inspection. Smart contracts are used in the proposed framework to store access control policies safely on the blockchain. The framework also penalizes users for attempting unauthorized access to the IoT resources. The smart contract was developed using Remix and deployed on the Ropsten Ethereum testnet. We analyze the performance of the smart contract-based access policies based on the gas consumption of blockchain transactions. Further, we analyze the system's security, usability, scalability, and interoperability performance.

Keywords: smart contract; blockchain; access control; Internet of Things; Ropsten test network; gas cost



Citation: Hasan, M.R.; Alazab, A.; Joy, S.B.; Uddin, M.N.; Uddin, M.A.; Khraisat, A.; Gondal, I.; Urmi, W.F.; Talukder, M.A. Smart Contract-Based Access Control Framework for Internet of Things Devices. *Computers* **2023**, *12*, 240. <https://doi.org/10.3390/computers12110240>

Academic Editors: Paolo Bellavista and Leandros Maglaras

Received: 24 July 2023

Revised: 12 November 2023

Accepted: 13 November 2023

Published: 20 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Internet of Things (IoT) effortlessly gathers and shares data from diverse embedded devices, sensors, and actuators. This inherent capability positions it as a promising network scenario, promoting efficient data exchange and interconnected functionality [1,2]. According to a recent study, the current count of IoT devices stands at almost 13.15 billion in 2023, with an anticipated increase to over 25.4 billion by 2030. This exponential growth underscores the expanding role and significance of IoT in our interconnected digital landscape [3].

IoT is prevalent in almost every aspect of life, such as healthcare, smart cities, and transportation [4,5]. For instance, by fastening wearables or sensors on patients, doctors can monitor their condition in real-time when they are away from the hospital. The Internet of Things can enhance medical care and avoid fatalities in high-risk patients by continuously monitoring specific metrics and sending automatic alerts on their vital signs. IoT offers potential solutions to address urban problems such as pollution, traffic congestion, and energy shortages. IoT applications include the smart home, self-driving cars, smart grid, IoT

retail shops, smart parking, smart supply-chain management, environmental monitoring, industrial internet, and e-voting, to name a few [6–8]. As the number of IoT applications increases, more critical information, including personal or confidential information, is produced. The most current IoT system can not guarantee trust and privacy for the data [9].

A rogue device might disrupt the IoT network's operation and result in disastrous outcomes. The IoT environment is battling problems with heterogeneity, integrity, resource limitations, availability, privacy, and security susceptibility [10]. In addition, authentication and access control are the first lines of protection because they only allow individuals with the necessary rights to access data [11,12]. To guarantee data security and integrity, secure IoT systems require reciprocal permission between IoT devices and other networks [13–15]. If not, these systems will be vulnerable to various security issues, such as unauthorized access, data theft, and data modification [16–19]. Due to the heterogeneous nature and distributed architecture of IoT networks, establishing authentication between diverse IoT devices involves complex and varied rules and regulations. Maintaining this with the aid of third-party authorities presents significant challenges, including issues of trust and potential bottlenecks. Further, there are several access control mechanisms in the state-of-the-art works. For example, usage control model (UCON), organizational-based access control (OrBAC), capability-based access control (CapBAC), role-based access control (RBAC), and attribute-based access control (ABAC) have been utilized in the literature [20]. role-based access control (RBAC) refers to managing user access to resources based on their roles [21]. The attribute-based access control (ABAC) is a logical access control paradigm that controls the access between subjects and objects by the properties of entries, operations, and related environments [22]. In conventional security systems, these access control mechanisms are implemented using centralized architectures, which are susceptible to single-point failures, scalability challenges, lower reliability, and reduced throughput. To address this issue, at present, researchers have sought blockchain technologies, which have only recently emerged, to successfully provide a solution to improve scalability, privacy, security, validity, and reliability. Blockchain is a decentralized platform where every transaction is carried out decentrally [23,24].

Patil et al. proposed a framework of access control using blockchain technology [25]. Further, Nayabe et al. [26] proposed a blockchain-based authentication mechanism for establishing secure communication between cars and shortening the time required for message transmission and verification. Similarly in [27], Bera et al. suggested a decentralized access control systems for the IoT environment, which allows mutual authentication between two surrounding drones and their corresponding ground station servers.

IoT systems require operation in a distributed fashion, with minimal delay to facilitate device interactions and deliver crucial services. Consequently, distributed security measures are essential to ensure the protection of these systems. Traditional security mechanisms, like authentication procedures, often fall short due to the centralized and non-scalable nature of IoT systems. For instance, an airborne drone transmitting time-sensitive data may need rapid authentication with multiple command stations in a distributed environment [28]. Most existing solutions fall short in addressing the emerging challenges in IoT. Many fail to fulfill key IoT attributes like usability, scalability, interoperability, security, and automation. To address these significant issues effectively, novel security and access control strategies in distributed frameworks are required. This paper introduces a blockchain-based access control framework to tackle these hurdles, aiming to enhance trust and facilitate broader adoption of IoT technology.

In this paper, we proposed a decentralized access control framework that incorporates a verification process for authorization. In our framework, a user does not get access to the IoT resources until the verification for authorization is successful. The system also penalizes users who attempt to gain unauthorized access to IoT resources. We store the access policies to be written using smart contracts. The smart contract was created with Remix and deployed on the Ropsten Ethereum testnet. We examine the performance of smart contract-based access controls based on blockchain regarding gas usage. Furthermore, we

assess the system's performance in terms of security, usability, scalability, interoperability, and automation.

Our main contributions are summarized below:

- We developed a smart contract-based authentication and access control mechanism for IoT. The framework is divided into three parts. The first part is to identify all the people and resources involved. The second part is to control access to network resources, while the third part is used to examine the behavior of users.
- We analyzed our framework using Slither to detect vulnerabilities in our smart contract code. We also tested the model on the Ropsten Ethereum test network and measured gas consumption. The price of this is then compared with a number of the current IoT authentication methods.

The remainder of the paper is arranged as follows. Section 2 provides a brief review of previous studies. Section 3 describe the proposed access control framework for IoT. Section 4 presents the findings of the experiments as well as a discussion. Finally, Section 5 concludes this study with future works. The acronyms that are used in this paper are listed in Table 1.

Table 1. List of acronyms used in the article.

Acronym	Full Form
ACP	Access Control Protocols
UCON	Usage Control Model
OrBAC	Organizational-Based Access Control
CapBAC	Capability-Based Access Control
RBAC	Role-Based Access Control
ABAC	Attribute-Based Access Control
RO	Resource Owner
AID	Allowance ID
ABI	Application Binary Interface
ECDSA	Elliptic Curve Digital Signature Algorithm

2. Related Works

This section summarizes the various blockchain-based authentication and access control options for IoT.

Ouaddah et al. [29] presented a token-based access control paradigm called “FairAccess”, which manages access policy efficiently and restricts policy reuse by deploying smart contracts. The authors employed public and private tokens to indicate user access rights, which may be transferred between peers. The token recipient must unlock the lock scripts to prove the token ownership. Though it is a brilliant concept to lock scripts for access control, the processing capacity of the locking scripts is rather limited.

Xu et al. [30] suggested a blockchain federated IoT access control system based on federal capacity. The architecture takes two IoT domains into consideration. For each area, the cloud elects the coordinator and transfers the decision-making process to the coordinators, which contributes to the system's scalability. The coordinator writes and registers blockchain policies. The procedure of verification of access rights is carried out in the IoT device using the local chain data synced with the blockchain network. Thus, certain IoT devices incur the cost of retaining local chain data, reducing the system's usability. Additionally, the compatibility of IoT devices and blockchain technology is not examined when it comes to synchronizing local chain data.

Hammi et al. [31] examined the blockchain concept's feasibility for solving different security challenges in IoT. The paper proposes a blockchain-based authentication system.

It enables decentralized authentication for IoT technology. The primary disadvantage of the suggested approach is that devices from one system cannot connect with devices from another system. As a result, it is inapplicable to a variety of dispersed IoT applications where communication between IoT devices belonging to various systems is crucial.

Han Liu et al. [32] designed and implemented an access control system named fabric IoT based on Hyperledger Fabric. In the proposed scheme, there were three smart contracts, namely policy contract (PC), device contract (DC), and access contract (AC). The authors implemented the ABAC policy management and ensured the access security of the device resources by implementing the smart contract application. This system utilizes a distributed architecture to manage the physical network's access control in a fine-grained and dynamic manner. However, the reliability and performance of the system is limited.

Using blockchain technology, Sivaselvan et al. [33] built an IoT access control system that uses capability-based authentication. A capability token is a digital representation of the access privileges granted to the device that holds it. The suggested architecture employs smart contracts to execute all actions, contributing to its scalability. However, no blockchain technology is included in IoT devices for authentication or access control. The essential connectivity between IoT devices and the blockchain network is achieved via interfaces that convert IoT-COAP messages to blockchain-compatible JSON-RPC messages and vice versa.

Khalid et al. [34] developed a decentralized authentication system for Internet of Things (IoT) devices that is suitable for a wide variety of scenarios. The mechanism is built on fog computing technology and the concept of a public blockchain. In general, the fog nodes belong to different people and may not be made by the same company, which makes it less safe. The elliptic curve digital signature algorithm (ECDSA) is utilized in this approach to generate public and private keys for devices and fog nodes. The issue identified in this work is that PoW consumes a lot of energy to validate each block.

Weizheng Wang et al. [35] introduced a smart contract token-based solution for decentralized access control in the Industrial Internet of Things (IIoT). While highlighting the use of the n th-degree truncated polynomial ring units (NTRU) for post-quantum encryption and a prototype platform for performance evaluation, certain limitations emerge. The paper lacks in-depth discussions on the token mechanism, security evaluation metrics, scalability considerations, and a clear distinction between the prototype and real-world implementations. Additionally, a more thorough comparative analysis with existing solutions is needed to comprehensively assess the proposed scheme's strengths and weaknesses in the context of IIoT access control.

Feifei Guo et al. [36] have proposed a domain attribute-based access control (DABAC) approach to address access control challenges in dynamic IoT environments. The proposed solution relies on an intelligent gateway for regional device management, which may introduce a single point of failure and potential scalability concerns. Additionally, the implementation on the Ethereum platform, while illustrating feasibility in a simulated smart medical scenario, raises questions about real-world scalability, transaction speed, and resource consumption. The effectiveness of DABAC in mitigating threats is asserted but requires substantiation through a more comprehensive analysis of potential drawbacks and comparative assessments with existing solutions.

An Internet of Things (IoT) access management strategy based on smart contracts was proposed by O.novo in [37]. It makes no attempt to integrate blockchain technology with IoT devices. In contrast, the necessary interactions between IoT devices and the blockchain are formed through management hubs, which serve as middlemen between the two technologies. The interface makes use of the Web3 JavaScript API to connect with the Ethereum nodes using RPC calls, as well as a CoAP library named node-coap5 to connect with the IoT devices. The scheme's usability, scalability, and interoperability are all strong characteristics. The security features, on the other hand, are restricted. There is no way to verify the legitimacy of the management hubs.

Xuanmei et al. [38] have presented a lightweight decryption-based access control mechanism based on fabric blockchain technologies. The authors have shown how to use fabric blockchain technologies to keep one's information secure. The blockchain's security mechanisms ensure that outsourcing decryption works successfully without requiring additional computation. However, they could not provide dynamic attribute management or automated smart contract features.

From the above related works, we can conclude that present state-of-the-art access control methods do not adequately address essential IoT attributes, including usability, scalability, interoperability, security, and automation. These criteria have been widely acknowledged in the literature as key factors contributing to the success of IoT solutions. Specifically, usability ensures a user-friendly experience, security addresses the protection of data and devices, scalability focuses on accommodating growth seamlessly, integrity ensures data reliability and accuracy, and automation emphasizes the efficiency of operations. In this paper, we tried to fill in this important research gap. We developed a new blockchain-based authentication and access management system for the Internet of Things. In addition, we deployed smart contracts to create a proof-of-concept version of the system.

3. Proposed Framework

This section summarizes the proposed system's general operation. The framework includes several servers and storage devices, as well as one or more IoT gateways and end-user devices that are all linked together via a P2P network. First, we describe the primary roles of the peers below.

- **Server:** Servers are software or hardware devices that accept and reply to requests that are sent via a network connection. The device that sends the request to the server and gets a reply from the server is referred to as a client. Mostly on Internet, the term "web server" refers to a computer networks system that accepts requests for web pages and then delivers those items to the client.
- **User device:** User devices include personal computers, laptops, smartphones, and smart watches, which allows users to access and use the services provided by the servers, as well as read data from and write information to storage devices.
- **Storage device:** A storage device is a sort of hardware, often known as storage media, that is capable of temporarily or permanently storing information. The storage medium is often used to store, transport, and extract data files. This can be used to store data inside or outside of a computer system, server, or any other type of computer.
- **IoT gateway device:** Gateways act as a wireless access portal, allowing IoT devices to connect to the internet. Gateways can be physical or virtual devices such as Bluetooth, WiFi, and Raspberry Pi.
- **IoT device:** IoT devices are physical components, such as sensors, actuators, gadgets, appliances, or machines, that are configured for a specific purpose and can transmit data via the internet or other networks. They can also be utilized for additional purposes. They can be integrated with, among other things, other mobile devices, industrial machinery, environmental sensors, and medical devices.

Our proposed methodology is segmented into three well-defined stages: initialization, access control protocol (ACP), and inspection. A comprehensive visual representation of our access control framework is provided in Figure 1. The initialization phase lays the foundation of the entire system by setting up trusted nodes and establishing secure communication channels.

The ACP phase forms the heart of our approach, effectively managing and enforcing access control policies that regulate the interaction with resources in an IoT setting.

In the inspection phase, we introduce a vigilant monitoring system. This phase is crucial for evaluating the conduct of subjects who seek access to resources, thereby ensuring continuous adherence to established norms.

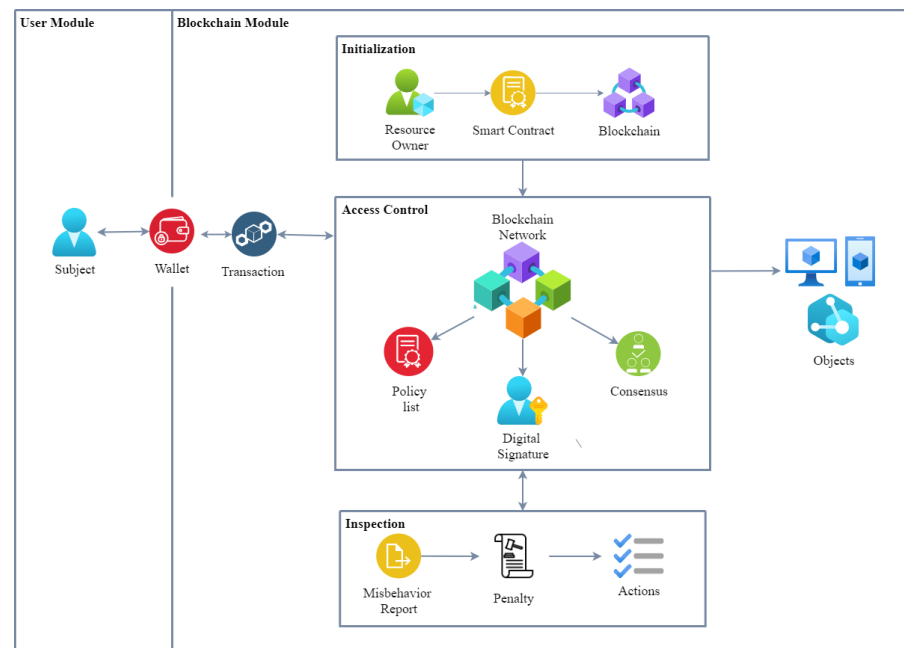


Figure 1. Overview of the proposed access control framework.

3.1. Initialization Phase

The initialization phase facilitates the identification of trusted nodes, linking them to the access control protocols (ACP) and their related functions. For integration into the blockchain network, all associated entities are required to create at least one unique account, distinguishable by a key pair. The public key from each user is utilized to formulate an account address, serving as an identifier within the system, contingent on blockchain network validation.

Key participants in our framework include the resource owner (RO), who holds ownership of a specific resource, nodes that garner trust from the RO, and individuals seeking to utilize the RO's resources for various objectives. The RO can employ access control protocols (ACPs) to establish access permissions for their resources. These ACPs, formulated by the RO, are subject to audit and are constructed based on the allowance ID (AID) of the subjects they safeguard.

The data presented in Table 2 are securely stored on the Ethereum blockchain, serving as a decentralized repository. This blockchain meticulously logs transactions carried out through smart contracts, inclusive of those formulated using Remix IDE in conjunction with Metamask. Every entry in Table 2 signifies a distinct transaction record on the blockchain, thereby fortifying the distributed and unalterable nature of the ledger.

Table 2. Illustration of the record maintained by initialization phase.

Allowance ID	Subject Address	ABI
0	0 × 786CabceC02C6C3C08b8F06ad72ca47240c0aB23	access control()
1	0 × 786CabceC02d6h7g0r06F06ad72bca47240c0aB12	access control()
2	0 × 786CabceC02C6C372ca47240C08b8F06adc0aB32	access control()
...

Once the identity details of users and IoT devices are authenticated, as depicted in Figure 2, they can be validated through their respective addresses. Consequently, the resource owner (RO) gains the autonomy to manage its devices and resources, eliminating dependence on external companies.

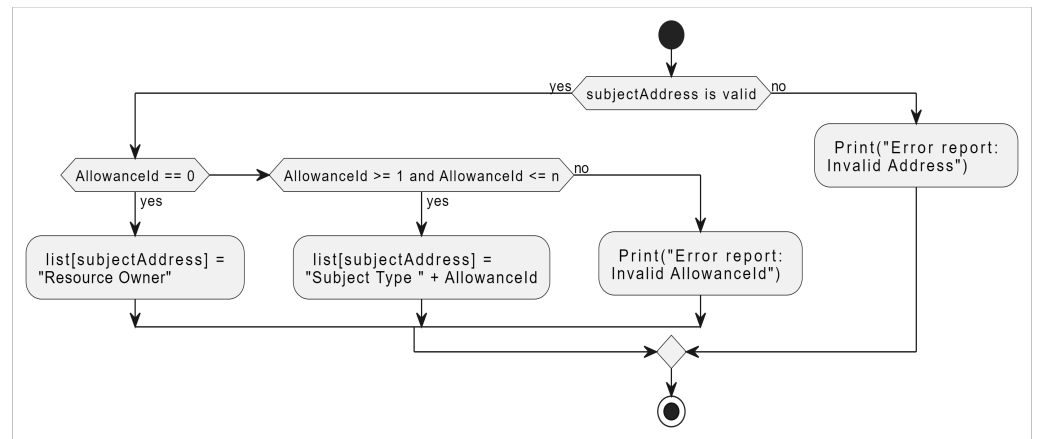


Figure 2. Illustration of Initialization phase.

Access control protocols (ACPs) are adept at authenticating a user's access rights by processing transactions sent from a trusted node. These transactions incorporate crucial elements such as the subject address, allowance ID (AID), and application binary interface (ABI). The procedural steps for this initialization phase are meticulously outlined in Algorithm 1.

Algorithm 1: Algorithm for initialization phase

Input: subjectAddress : The Address of the Entity or User, AllowanceId (AID): The Allowance ID indicating the Subject's Role or Type.

Output: list[subjectAddress]: The assignment of the subject's role or type in the access control system

```

if subjectAddress is valid then
    // Check if AllowanceId is within the valid range.
    if AllowanceId = 0 then
        // Assign subjectAddress as "Resource Owner".
        list[subjectAddress] = "Resource Owner"
    end
    else if AllowanceId ≥ 1 and AllowanceId ≤ n then
        // Valid AllowanceId, proceed with assignment.
        list[subjectAddress] = "Subject Type" + AllowanceId
    end
    else
        // Invalid AllowanceId, report an error.
        Print("Error report: Invalid AllowanceId")
    end
end
else
    // Invalid subjectAddress, report an error.
    Print("Error report: Invalid Address")
end

```

Explanatory Notes on Algorithm 1

AID (allowance ID) represents a numeric identifier associated with subjects in the access control system.

- The range of AID is from 0 to n, offering a variety of possibilities for categorizing subjects based on their roles or types in the system.

- AID 0 is designated for the resource owner (RO), indicating the primary entity with ownership and control over the resources.
- AID 1 to n is reserved for different subject types or roles, each serving a distinct purpose within the access control framework.
- If a subject's AID falls outside this specified range (0 to n), the system generates an error report, signaling an "Invalid AllowanceId". This mechanism ensures that only valid and predefined AID values are accepted, maintaining the integrity and security of the access control system.

3.2. Access Control Protocol Phase

Our proposed decentralized access control mechanism incorporates a two-factor authentication process to enhance security.

- **Initial Authorization Check:** The process begins by determining if the subject is authorized to make an access request. This involves checking various criteria to validate the access request.
- **Device Address Verification:** After passing the initial step, the system checks if the subject's device address is registered. If the device is already registered, it can access the specified resource without needing to sign a message. If the device is not registered, the subject must sign a message using the chosen device that is currently logged in.
- **Identity Verification through Signed Message:** The system retrieves the identity of the signer from the signed message. The hash of this signed message is computed using the private key of the requester's Ethereum account.
- **Final Authentication and Device Registration:** Upon confirming the signer's identity, the system verifies that the requester is indeed the legitimate user of the requested resources. The new device is then registered for future use, ensuring streamlined access in subsequent requests.
- **Dynamic Access Control through Access Control Protocol List (ACP):** For managing access requests from peers, we assume one access control (AC) implements the agreed-upon access control mechanisms for each subject-related object. Each ACP list as shown in Figure 3 performs two crucial functions:
 - It checks not only the established policies for the subject but also monitors the subject's behavior during access attempts.
 - The ACP dynamically evaluates policies for a subject, ensuring compliance with predefined rules. At the same time, it scrutinizes the subject's real-time behavior, providing an adaptive and responsive layer of control.

By incorporating these steps, our system effectively confirms the identity of the requester and dynamically adapts to ensure compliance and security.

Policy management: Our Ethereum smart contracts are equipped with interfaces and functions that facilitate policy-related activities through transaction execution. This setup allows our proposed access control protocol (ACP) to efficiently handle the various functions necessary to operate these application binary interfaces (ABIs).

- **Adding a Policy:** If there is an agreement between a subject-object and a newly-deployed resource to add an access control policy, the resource owner initiates this process. They do so by sending a message to trigger the `addpolicy()` function in the relevant ACP.
- **Updating a Policy:** To modify an existing access control policy, the owner can send a message to call the `updatepolicy()` function.
- **Changing Location:** The `locationupdate()` function is available for the owner to change the location specified in an ACP policy list. To do this, the policy's identity details are needed.
- **Adjusting Time Range:** The policy creator can also alter the policy's time range using the `timerange()` method.

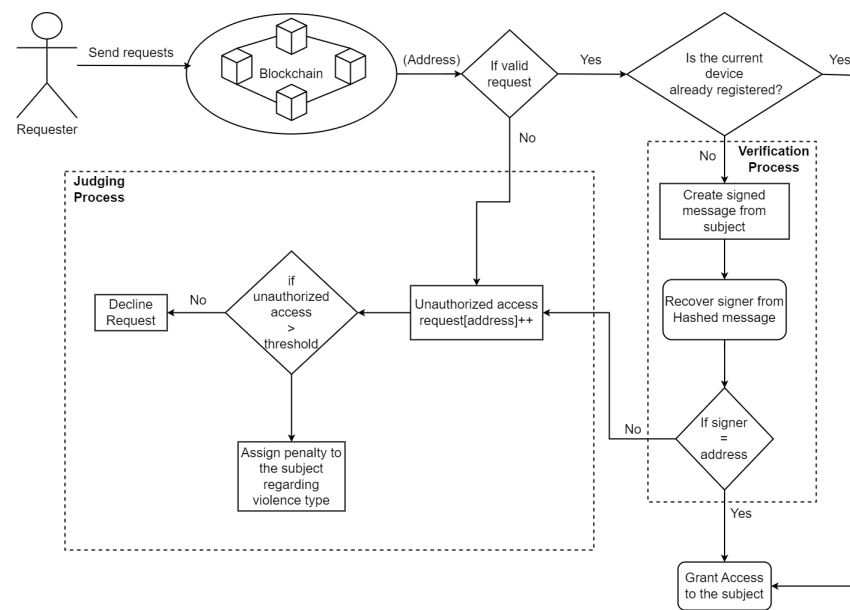


Figure 3. Illustration of Access Control Protocol phase.

To illustrate these processes more clearly, we have Table 3. This table demonstrates an owner's policy list, breaking down categories like subject (Ethereum address), resource, action, location, time range, and permission. It provides a concise overview of the intricate access control policies managed through Ethereum smart contracts in our system. The table ensures that each step, from the deployment of the smart contract to the implementation of the policy, is transparent and effectively enforced. The following segments of the policy list are examined in greater detail:

- **Subject:** Identifies the subject through a unique identifier, such as their Ethereum address.
- **Resource:** Specifies the name of the resource that needs to be included in the access control process.
- **Action:** Outlines the permissible actions on the specified resource, which could include viewing, downloading, reading, or writing.
- **Location:** Provides details about the location from which the subject can access the resource.
- **Time Range:** Allows the resource owner (RO) to set a specific time range during which the resource can be accessed.
- **Permission:** Indicates whether a particular pairing of resource and subject has been granted permission for certain actions, like allowing or denying access.

Table 3. Illustration of the owner's policy list.

Resource	Subject	Location	Time Range	Action	Permission
File A	User 2	Location A	02:05–03:10	Read	Deny
File B	User 4	Location B	14:10–16:20	Download	Allow
File C	User 9	Location D	20:24–22:25	Write	Deny
File R	User 5	Location G	05:30–05:50	View	Allow

Access Control: The access control policy (ACP) serves as a mechanism for both the resource owner and the subject to manage access interactions between the subject and an object. It is assumed that both parties are knowledgeable about all available access control strategies. Utilizing this access behavior interface (ABI), they can obtain necessary access control information, which then provides an outcome regarding access and any associated penalties. When the subject invokes this function to authenticate a recent access request, the system initiates a verification process to determine its legitimacy. If a potential violation is detected, the ACP prompts the inspection phase by instructing the GetViolationHistory ABI to enact a violation response. This inspection phase then delivers a verdict on any penalty related to the violation. Once both static and dynamic verifications are satisfactorily completed, the access request is granted.

Explanatory Notes on Algorithm 2

Algorithm 2 is designed for policy management within a context where certain parameters (parameter1 and parameter2) and contextual information such as RequesterAddress, message, and currentDevice play a crucial role. The algorithm aims to evaluate access requests and, if needed, assign penalties for violent actions. Here is a breakdown of the algorithm's steps:

1. Condition Checking:
 - The algorithm begins with a conditional check on parameter1 and parameter2. This check acts as a decision point, allowing the algorithm to follow different paths based on the values of these parameters.
 - Parameter1 and parameter2 represent specific conditions or attributes relevant to the access request scenario. The choice of X and Y as comparison values may reflect specific requirements that trigger particular actions.
2. Device Registration:
 - If the condition (parameter1 = X and parameter2 = Y) is met, the algorithm proceeds to check the registration status of the currentDevice.
 - The process involves creating a hash of the message (msgHash) to ensure data integrity during digital signature verification.
 - The algorithm obtains a signature (sig) from the selected logged-in device using Ethereum's signing mechanism.
 - Signature verification (verify) ensures that the signature corresponds to the provided message, confirming the identity of the signer.
 - If the signer's address (recoverSigner) matches the requester's address (RequesterAddress), the currentDevice is registered for future use.
3. Access Request Handling:
 - After registration, or if the device is already registered, the algorithm checks if the signer's address matches the requester's address. This step is crucial for validating the authenticity of the access request.
 - If the addresses match, the algorithm successfully registers the new device for future use and returns true, indicating a successful access request.
 - If the addresses do not match, the algorithm enters the judgePhase. This phase handles penalties for bad access requests and returns false to signify an unsuccessful access attempt.

Algorithm 2: Algorithm for policy management

Input: *parameter1* : *AParameterValue*, *parameter2* : *AnotherParameterValue*,
RequesterAddress : *TheAddressoftheRequester*,
message : *AMessageReceived*,
CurrentDevice : *TheDeviceMakingtheRequest*

Output: Request Evaluation and Penalty Assignment for violence

```

if parameter1 = X and parameter2 = Y then
  if currentDevice is not registered yet then
    msgH ← msgHash(message); // Calculate the message hash for
    digital signature verification
    sig ← ethereumSignedMessage(msgH); // Obtain the signature from
    the selected logged-in device
    recoverSigner ← verify(sig); // Verify the signature and retrieve
    the signer's address
    if recoverSigner = RequesterAddress then
      registerNewDevice(currentDevice); // Register the new device
      for future requests
      return true; // Request is approved
    else
      judgePhase(RequesterAddress); // Assign a penalty for a bad
      access request
      return false; // Request is denied
    end
  else
    return true; // Device is already registered, and the request is
    approved
  end
else
  judgePhase(RequesterAddress); // Assign a penalty for a bad access
  request
  return false; // Request is denied
end

```

3.3. Inspection Phase

The inspection phase employs a misbehavior judging method upon receiving a report from an ACP regarding possible misbehavior. This method assesses the nature of the subject's inappropriate behavior and determines the appropriate penalty. The decision regarding the penalty may take into account the subject's misbehavior history. Consequently, the inspection phase is responsible for maintaining a comprehensive record of misbehavior histories for all individuals. Once the penalty is determined, the inspection phase proceeds to forward its judgment to the ACP for further action. To provide further clarity on the usage of penalties, let us illustrate how this phase maintains a detailed list of misbehavior incidents for each individual who has engaged in improper actions, as depicted in Table 4. Each record has a set of fields:

- **Peer:** The person who was affected by the misbehavior
- **Misbehavior:** The facts of the misbehavior, or what happened
- **History:** The number of misbehavior occurrences;
- **Penalty:** The punishment for misbehavior action.

The inspection phase manages the records using the following function.

- **GetViolationHistory():** In order to decide on an appropriate penalty, this function examines the behavior information obtained from the ACC. Following the report,

this function judged the misconduct of subjects, determined the penalties for subjects based on the subject's violation history, and sent the penalty decision back to the ACC that reported the misbehavior. The subject's violation record is also updated by this function.

Table 4. Violation records maintained by Inspection phase.

Subjects	Violation Type	History	Penalties
User 4	Too frequent access	3	Blacklisted
User 2	Unmatched location	3	Access denied
User 5	Incorrect access request	4	Access denied
User 1	Too frequent access	5	Blacklisted
...

Table 4, illustrates an example of a document used to keep track of a user's violation. To illustrate such situations, let us assume User 4's repeated access requests to the RO's ACP, as indicated in Table 4. One way to deal with this situation could be to blacklist the person involved. In the case of User 2, the RO returns an access denied response since the user's location is not included in the list. Finally, since User 5's access requests are not included in the policy list, a rejected result is applied to handle this case. Additionally, Algorithm 3 provides a detailed procedure for handling access violations as exemplified in Table 4.

Algorithm 3: Algorithm for judge phase

Input: *RequesterAddress, location, deviceName*
Output: Misbehavior Inspection
if *access_control_policy(req_add, location, device) = true* **then**
 | *result* \leftarrow *true*;
else
 | *result* \leftarrow *false*;
 | *unauthorized_access[req_add]* ++;
 | **if** *unauthorizedAccess[RequesterAddress]* \geq *threshold* **then**
 | *unauthorizedAccess[RequesterAddress]* \leftarrow 0;
 | *violationHistory[count ++]* \leftarrow *RequesterAddress*;
 | *penalty[RequesterAddress]* \leftarrow *penaltyID*;
 | **end**
end
return result;

4. Implementation and Results

4.1. Implementation

This implementation section functions as a prototype, offering a scaled-down demonstration of our concept. We have integrated just two devices into the system/framework to showcase its core functionalities. Our development process focused on blockchain-related components and was executed using Remix, without the creation of a graphical user interface. The entirety of this development process was managed within the integrated development environment (IDE), proving to be highly effective for our prototype.

To exemplify the potential applications of our framework, we present a representative use case. In this scenario, we demonstrate the remote control of light-emitting diodes (LEDs) as the target resource. The specific device of interest is an LED equipped with remote control capabilities. This LED is constructed using a NodeMCU board, which is

built upon the ESP8266 platform, facilitating item connectivity and data transmission via the Wi-Fi protocol.

The device is accessible remotely through a wireless LAN (WLAN) connection. The interactions of various components within this case study are visually represented in Figure 4.

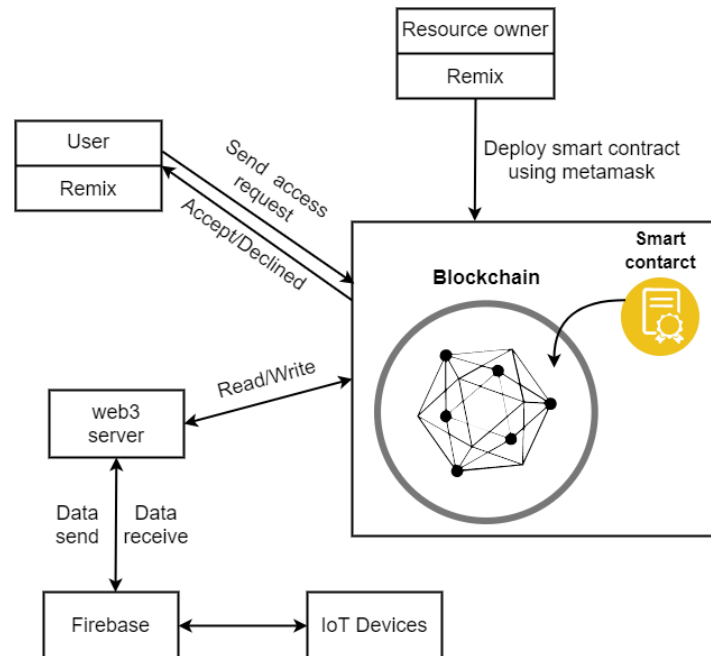


Figure 4. The interaction of several components in the case study.

We created a two-tiered access hierarchy on a single-board device. For the time being, we will refer to them as LEVEL1 and LEVEL2. At LEVEL1 access depth, a real user can only see the LED state of our target device. In addition to reading the device's status, a valid user with LEVEL 2 accessibility can change the LED's status and turn it off. We begin by introducing the hardware and software utilized in the research and then demonstrate how the access control framework is implemented. Finally, we present the outcomes of certain experiments. Algorithms 4 and 5 define the access control procedures for LEVEL1 and LEVEL2, respectively.

Algorithm 4: Algorithm for LEVEL1 access level

Input: *parameter1, parameter2, Requester Address*

Output: LED status

if *accessibility(Requester Address) = LEVEL1* **then**

 | **return** *LED_STATUS*

else

 | **return** "Access Request Denied"

end

As shown in Figure 5a, the resource owner can change the user/role subject and set how far it can get into the system. This is carried out with the set role and set state ABI. Figure 5b displays how a requester might send an access request for a certain resource by giving parameters such as the resource's location, account address, and so on. The requester must additionally sign a message for verification purposes, as presented in Figure 5d, and the system will then extract the signer's identity from the message that was signed. This enables the system to confirm the identity of the requester.

According to the case study, the light-emitting diode, considered a resource of the owner, is controlled via a toggle menu. When the owner of a resource grants the user access to that resource, the resource is under the control of the user. Let's say that a legitimate user wishes to alter the light's intensity. Figure 5c shows toggle options that allow the user to change the current state of an LED after it has been correctly validated during verification. In Figure 5e, a web3.js application shows a successful attempt to change this resource's state.

(a) Set role

(b) Access request

(c) A toggle for certified requester

(d) Sign message request

Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717
Ready!	(index):717
LED Status: on	(index):717

(e) LED resource status on web3.js

Figure 5. Implementation results.

The Javascript files on the web3-enabled server continuously check the status value that is stored at the deployed smart contract. If any change happens, i.e., LEVEL2 access is triggered, it receives the update and makes a change to the Firebase database LED status. On the other hand, the IoT device keeps listening to the Firebase database. If it notices any change, it makes the change in real to the connected LED.

Algorithm 5: Algorithm for LEVEL2 access level

Input: parameter1, parameter2, RequesterAddress

Output: LED status toggle report

```

if accessibility(RequesterAddress) = LEVEL2 then
  if LED_STATUS = "ON" then
    LED_STATUS ← "OFF";
    return "LED SET TO OFF";
  else
    LED_STATUS ← "ON";
    return "LED SET TO ON";
  end
else
  return "Access Request Denied";
end

```

- **Hardware and Software**

As depicted in Figure 6, our study used two laptop computers (Dell Vostro 356 500, Asus VivoBook S14) and one NodeMCU board-integrated LED board. The NodeMCU board is selected due to its compatibility with IoT projects, cost-effectiveness, strong open-source community support, integrated Wi-Fi, and scalability. Table 5 lists the technical specifications of these devices, and Table 6 indicates the software tools used for the case scenario. The laptops are the user devices in the system, while the LED board is the object. We took into account the issue of access control between the subjects and the objects. We used the Remix integrated development environment (IDE), a browser-based IDE for Solidity (the programming language for developing smart contracts) to write and compile the smart contract. Additionally to this, we made use of web3.js (the official Ethereum JavaScript API) at the object level to communicate with the associated geth client via HTTP connections to deploy and also monitor the state of our code (i.e., the results of the access control). Additionally, on the subject side, the web3 was set up to communicate with the geth using transactions to submit access requests to the ACP and to get access control responses from the smart contract.

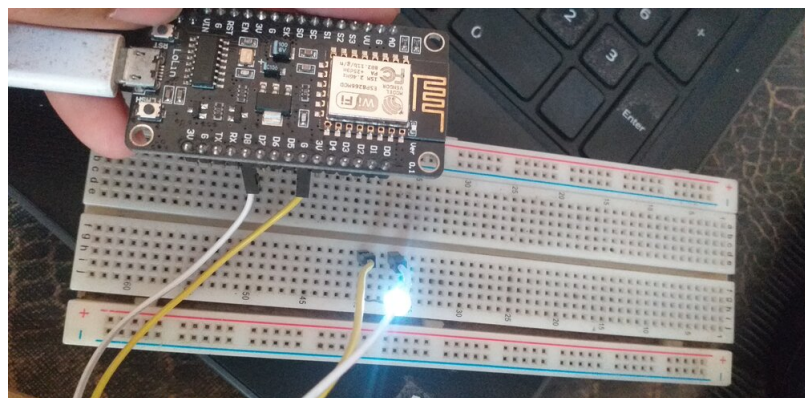


Figure 6. Hardware used in the case study.

Table 5. Specification of Hardware Devices.

Device	CPU	Operating System	Memory	Hard Disk
Dell vostro 5000	Intel Core i3-7100U, 2.40 GHz	Ubuntu 20.04.4	4 GB	1 TB
Asus VivoBook S14	Ryzen 5 4500u, 4 GHz	Windows 10 Home (64 bit)	8 GB	512 GB
NodeMCU	Tensilica 32-bit RISC cpu, 80 MHz	—	64 kb	flash memory 4 MB

Table 6. Software Tools Used for development.

Blockchain Platform	Ethereum 2.0
Development Environment	Remix IDE v0.11.0
Smart Contract Language	Solidity 0.8.19
Web Integration	Firebase 8.8.1
Security Analysis	Slither 0.9.0
Operating System	Linux 5.18

4.2. Security Requirements and Implementation

In this section, we will analyze how our suggested approach aligns with a range of security prerequisites. The choice of the following security prerequisites stems from a thorough evaluation of potential threats and the overarching objective of guaranteeing the resilience of our proposed method. Each security measure has been carefully selected to tackle distinct concerns and fortify the system's overall security stance.

- **Integrity:** To ensure system integrity, data are signed using the sender's private key before being sent to the receiver. This is performed in order to generate a data packet that is compatible with Ethereum's ECDSA algorithm. Finally, the final packet comprises the data, a hash of the data, and a signature issued by the sender. The receiver validates it using the signer's address and the hash of received data [39].
- **Identification:** All devices entering the system must have device identification (ID) in order to meet this security standard. For all devices registered in the system, an ID is assigned to identify the receiving device, and this ID must be provided to other devices in the system for communication. So the device may extract the sender's ID and know the system it belongs to.
- **Mutual authentication:** The suggested mechanism includes an allowance ID (AID), which was addressed in detail in Section 3. The AID is created by the subjects' private key. The AID of each device in the system is genuine. This boosts the confidence of other nodes in communicating with one another [40].
- **Scalability:** Our approach uses a public blockchain and a peer-to-peer network. P2P networks are widely considered to be one of the greatest ways to achieve scalability at big scales.
- **Non-repudiation:** Each transaction in the system is signed using the private key associated with it. There is no way to dispute that a transaction was carried out; hence, the sender cannot deny it (repudiate).
- **Spoof attack:** Device ID, system ID, and private key are required for a successful spoof identity attack on a target device. Even if the attacker obtains system IDs and device IDs, the intruder will still require the private key in order to fake the device identity [41].
- **Sybil attack:** To carry out a Sybil assault, the attacker must construct bogus identities. Multiple IDs are not permitted in the proposed process, and thus each device will only have one ID that is registered with the system. The private key associated with

a device is used to encrypt and sign the message. This eliminates the possibility of creating false identities into the system [42].

- **DoS/DDoS attack:** Blockchains' decentralized architecture makes systems resistant to denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. Transactions are also expensive, and thus an attacker is less likely to spend money by sending several transactions. Blockchain technologies such as Ethereum also have transaction fees that depend on the size of the transaction packets that are sent [43]. In summary, these security requirements are rooted in a thorough understanding of potential vulnerabilities and the strategic application of measures to address them. This approach ensures a robust and resilient security framework for the proposed technique.

4.3. Smart Contract Analysis Using Slither

Modifications to the smart contract code are only feasible prior to its deployment on the main network. Once deployed on the main network, any changes become irreversible. The vulnerability of smart contracts due to weak coding has led to instances of malicious actors pilfering substantial sums of money in the past. Hence, it is imperative to conduct a comprehensive examination of smart contracts before their implementation.

To this end, we employed Slither, a static analysis framework designed for smart contracts, to identify vulnerabilities within our code. This tool can also facilitate code optimization and scrutiny. Following the analysis of our contract code with this tool, the results are depicted in Figure 7. These results indicate that our code exhibits two low-level issues, with zero medium and high-level issues identified. Slither attributed the low-level issues to problems related to built-in symbol shadowing. Importantly, these issues have no adverse impact on the code's efficiency or the security of our smart contract.

Number of low issues: 2
Number of medium issues: 0
Number of high issues: 0

Name	# functions	ERC5	ERC20 info	Complex code	Features
Access_Control_Protocol	16			Yes	Ecrecover Assembly

. analyzed (1 contracts)

Figure 7. Summary of our smart contract functions and issues.

4.4. Costs Evaluation and Comparison

Every time a smart contract function is called, a fee is incurred to compensate the mining node for processing and recording the transaction on the blockchain. In Ethereum, this cost is denoted in terms of “gas”, and it can be procured from mining nodes in exchange for Ether. It is essential to bear in mind that while gas represents a consistent expense for executing actions on the blockchain network, Ether is a volatile digital currency used to settle charges for the network's resources.

To evaluate and compare costs, we constructed prototypes of our proposed smart contracts using the Solidity programming language. These prototypes were subsequently deployed on the Ropsten Ethereum testnet to assess their functionality. As of 25 March 2023, we observed that the average gas value amounted to 0.0000002036. The following section will provide a comprehensive breakdown of the specific expenses associated with various operations.

In the provided Figure 8 and detailed in Table 7, the subject registration process incurs a fee of 0.000956 ETH and consumes 46,935 gas on the test network. Similarly, a request for access is associated with a cost of 0.0014 ETH and a gas consumption of 68,272 on the test network. Additional details regarding the costs of various operations can be found in the accompanying table.

Table 7. Costs of the different operations.

Operation	Gas Used	Transaction Fee (ETH)
Contract deployment	2,758,894	0.05617
Subject registration	46,935	0.000956
Access request	68,272	0.0014
Recover signer	23,918	0.000486
Message signing	22,753	0.0004632
Access control protocol	127,055	0.0025
Accessibility check	26,659	0.000542
Violation history check	23,962	0.000493

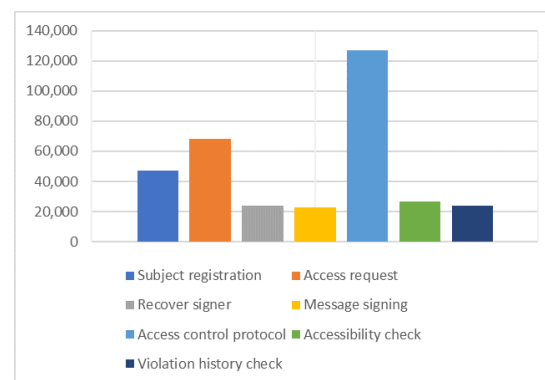
**Figure 8.** Costs of multiple functions on smart contract.

Figure 9a illustrates the one-time constant costs of registration functions on smart contracts in comparison to various alternative ways. For data sharing, T. Sultana et al. [44] presented an Ethereum smart contract-based access control system. As a consequence of their experiments, they discovered that it takes 130,000 gas to create a simple registration policy function on the Ethereum blockchain. N. Sivaselvan et al. [33] suggested a blockchain-based scheme for authentication and capability-based access control in the Internet of Things (IoT) environment. It takes 60,785 gas to register on the Ethereum network. Our proposed framework, on the other hand, consumes only 46,935 gas on the Ropsten testnet, which is significantly less than the other two solutions.

Figure 9b demonstrates how much it costs to track misbehavior on our smart contract versus how much it costs to utilize alternative ways. The inspection phase collects information concerning the subject's misbehavior, as well as the time of the offense, in order to establish an appropriate punishment.

The one-time constant costs of access control functions on smart contracts are depicted in Figure 9c in contrast to several alternative approaches. S. Y. A. Zaidi and colleagues suggested an attribute-based access control for the Internet of Things (IoT) that relied on blockchain and smart contracts. According to their testing results, it takes 820,457 gas to implement an access control function in the Ethereum blockchain. On the other hand, our suggested framework consumes only 292,619 gas on the Ropsten testnet, which is significantly less than the other three solutions.

Figure 9d compares the contract deployment and overall cost of various authentication and access control approaches. The present approach's overall deployment costs exceed the default 3,000,000 gas in the remix. However, the proposed method is within the default gas limit of 3,000,000 gas in remix and considerably less than other existing frameworks. For example, attribute-based access control for IoT systems developed in [45] required about 5 million gas, nearly double the amount required by our approach. Based on the costs, our proposed framework is clearly more lightweight than existing solutions.

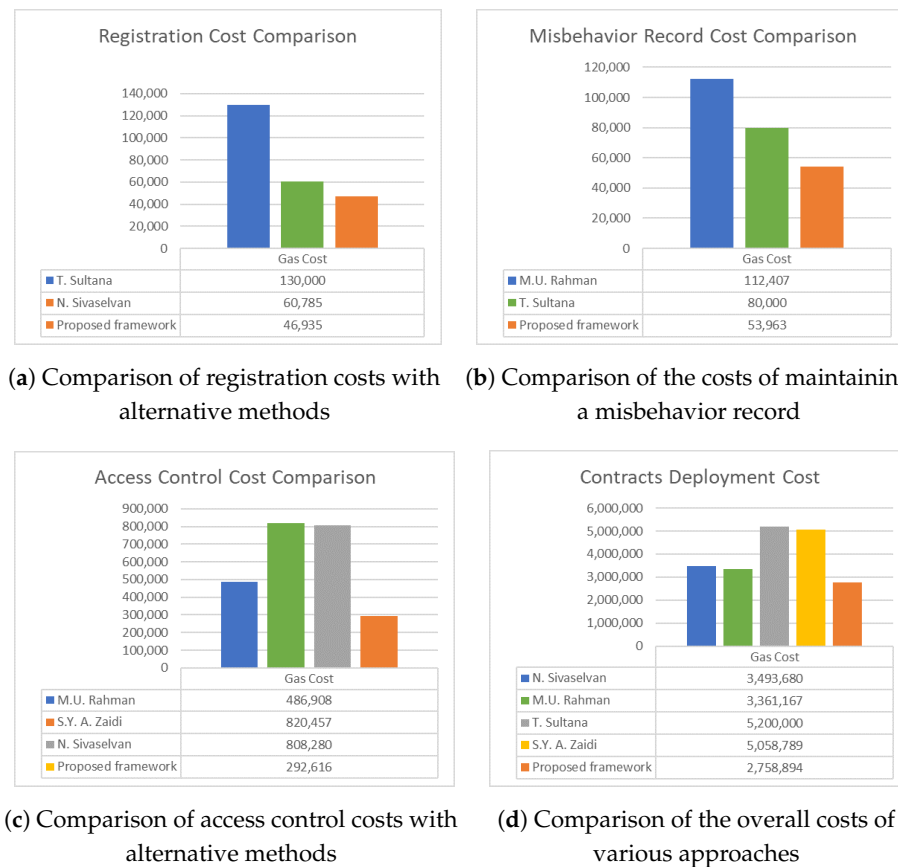


Figure 9. Cost Comparison.

Table 8 shows a comparison of the several properties of various authentication and access control approaches. It is noticeable that current state-of-the-art methodologies fail to reach the essential IoT attributes of usability, scalability, interoperability, security and automation as discussed on Section 2. The proposed technique, on the other hand, addresses all of these. As a result, as compared to current state-of-the-art methodologies, the proposed methodology is better suited for IoT.

Table 8. Comparison with the state-of-art approaches.

Paper	Usability	Security	Scalability	Interoperability	Automated
[29]	✓	✓	x	✓	x
[30]	x	x	✓	✓	✓
[31]	✓	✓	x	✓	✓
[32]	x	✓	✓	x	x
[33]	✓	x	✓	x	✓
[34]	✓	x	✓	✓	✓
[35]	✓	x	x	✓	✓
[36]	✓	✓	x	✓	✓
[37]	✓	x	✓	x	✓
[38]	✓	✓	x	✓	x
Proposed Scheme	✓	✓	✓	✓	✓

5. Conclusions

In the realm of IoT systems, users anticipate seamless operation within a distributed environment, characterized by minimal latency. This enables IoT devices to securely exchange time-sensitive data among themselves. In response to these demands, this paper introduces a smart contract-based access control system tailored for IoT scenarios. This decentralized approach to access control tackles the trust issue and enhances the overall stability of the system.

Our proposed access control system incorporates a two-factor verification process. The initial step assesses the subject's authorization to initiate the access request, while the second step validates the requester's identity. Smart contracts were deployed on the Ropsten test network, affirming the feasibility of our approach. Additionally, we utilized Slither, a security analysis tool, to identify vulnerabilities and security concerns within our smart contracts. The results of the security analysis confirm the practical safety of our scheme.

Looking ahead, we envision several avenues for future exploration and expansion:

- **Blockchain Integration:** We are committed to deeper integration with emerging blockchain technologies, such as sharding and layer 2 solutions, to enhance the performance and efficiency of our access control system.
- **AI and Machine Learning:** By harnessing the capabilities of AI and machine learning, we aim to bolster our security measures through advanced anomaly detection and behavior analysis within the IoT ecosystem.

In conclusion, our smart contract-based access control system is not only a solution for the present but a gateway to a future brimming with innovation and exploration in the realm of IoT security.

Author Contributions: M.R.H. and S.B.J.: Data curation, methodology, software, writing—original draft preparation; M.N.U., A.A. and M.A.U.: conceptualization, supervision, methodology, formal analysis, visualization, writing—reviewing and editing; A.K., I.G. and W.F.U.: visualization, investigation, validation, writing—reviewing and editing; M.A.T.: investigation, validation, visualization, writing—reviewing and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by Deakin University and the Air Force Office of Scientific Research under award number FA2386-23-1-4003.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors have no conflict of interest to declare that are relevant to the content of this article.

References

1. Iqbal, W.; Abbas, H.; Daneshmand, M.; Rauf, B.; Bangash, Y.A. An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security. *IEEE Internet Things J.* **2020**, *7*, 10250–10276.
2. Singh, P.; Khari, M. Necessity of Time Synchronization for IoT-Based Applications. In *Internet of Things: Technological Advances and New Applications*; Apple Academic Press: Cambridge, MA, USA, 2023; p. 285.
3. Danladi, M.; Baykara, M. Low Power Wide Area Network Technologies: Open Problems, Challenges, and Potential Applications. *Rev. Comput. Eng. Stud.* **2022**, *9*, 71–78. [[CrossRef](#)]
4. Santos, R.; Eggly, G.; Gutierrez, J.; Chesñevar, C.I. Extending the IoT-Stream Model with a Taxonomy for Sensors in Sustainable Smart Cities. *Sustainability* **2023**, *15*, 6594.
5. Malik, H.; Anees, T.; Faheem, M.; Chaudhry, M.U.; Ali, A.; Asghar, M.N. Blockchain and Internet of Things in Smart Cities and Drug Supply Management: Open Issues, Opportunities, and Future Directions. *Internet Things* **2023**, *23*, 100860.
6. Espinosa, Á.V.; López, J.L.L.; Mata, F.M.; Estevez, M.E.E. Application of IoT in healthcare: Keys to implementation of the sustainable development goals. *Sensors* **2021**, *21*, 2330.
7. González-Zamar, M.D.; Abad-Segura, E.; Vázquez-Cano, E.; López-Meneses, E. IoT technology applications-based smart cities: Research analysis. *Electronics* **2020**, *9*, 1246.
8. Uddin, M.A.; Stranieri, A.; Gondal, I.; Balasubramanian, V. A survey on the adoption of blockchain in iot: Challenges and solutions. *Blockchain Res. Appl.* **2021**, *2*, 100006.

9. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and Security: Challenges and Solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
10. Yaacoub, J.P.A.; Noura, H.N.; Salman, O.; Chehab, A. Ethical hacking for IoT: Security issues, challenges, solutions and recommendations. *Internet Things Cyber-Phys. Syst.* **2023**, *3*, 280–308.
11. Alharbi, A. Applying Access Control Enabled Blockchain (ACE-BC) Framework to Manage Data Security in the CIS System. *Sensors* **2023**, *23*, 3020.
12. Gupta, D.S.; Mazumdar, N.; Nag, A.; Singh, J.P. Secure data authentication and access control protocol for industrial healthcare system. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 853–4864. [CrossRef] [PubMed]
13. Taherdoost, H. Security and Internet of Things: Benefits, Challenges, and Future Perspectives. *Electronics* **2023**, *12*, 1901.
14. Vignesh Saravanan, K.; Jothi Thilaga, P.; Kavipriya, S.; Vijayalakshmi, K. Data Protection and Security Enhancement in Cyber-Physical Systems Using AI and Blockchain. In *AI Models for Blockchain-Based Intelligent Networks in IoT Systems: Concepts, Methodologies, Tools, and Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 285–325.
15. Rao, P.M.; Deebak, B. A Comprehensive Survey on Authentication and Secure Key Management in Internet of Things: Challenges, Countermeasures, and Future Directions. *Ad Hoc Networks* **2023**, *146*, 103159.
16. Mishra, S. Exploring the Impact of AI-Based Cyber Security Financial Sector Management. *Appl. Sci.* **2023**, *13*, 5875. [CrossRef]
17. Kafi, M.A.; Akter, N. Securing Financial Information in the Digital Realm: Case Studies in Cybersecurity for Accounting Data Protection. *Am. J. Trade Policy* **2023**, *10*, 15–26.
18. Duggineni, S. Impact of Controls on Data Integrity and Information Systems. *Sci. Technol.* **2023**, *13*, 29–35.
19. Bandari, V. Enterprise Data Security Measures: A Comparative Review of Effectiveness and Risks Across Different Industries and Organization Types. *Int. J. Bus. Intell. Big Data Anal.* **2023**, *6*, 1–11.
20. Hussein, D.; Bertin, E.; Frey, V. Access control in IoT: From requirements to a candidate vision. In Proceedings of the 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, France, 7–9 March 2017; pp. 328–330. [CrossRef]
21. Ouaddah, A.; Mousannif, H.; Elkalam, A.; Ouahman, A. Access control in The Internet of Things: Big challenges and new opportunities. *Comput. Netw.* **2016**, *112*, 237–262. [CrossRef]
22. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Friedman, A.R.; Lang, A.J.; Cogdell, M.M.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K.; et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST Spec. Publ.* **2013**, *800*, 1–54.
23. Dai, H.N.; Zheng, Z.; Zhang, Y. Blockchain for Internet of Things: A survey. *IEEE Internet Things J.* **2019**, *6*, 8076–8094.
24. Singh, S.; Hosen, A.S.; Yoon, B. Blockchain security attacks, challenges, and solutions for the future distributed iot network. *IEEE Access* **2021**, *9*, 13938–13959. [CrossRef]
25. Patil, P.; Sangeetha, M.; Bhaskar, V. Blockchain for IoT access control, security and privacy: A review. *Wirel. Pers. Commun.* **2021**, *117*, 1815–1834. [CrossRef]
26. Nayab, A.; Javaid, N. An Efficient Distributed Data Communication Framework Using Blockchain for Vehicle-to-Vehicle Communication. Available online: https://www.researchgate.net/publication/334626508_An_Efficient_Distributed_Data_Communication_Framework_Using_Blockchain_for (accessed on 1 November 2023).
27. Bera, B.; Chattaraj, D.; Das, A.K. Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment. *Comput. Commun.* **2020**, *153*, 229–249. [CrossRef]
28. Mohsan, S.A.H.; Khan, M.A.; Noor, F.; Ullah, I.; Alsharif, M.H. Towards the unmanned aerial vehicles (UAVs): A comprehensive review. *Drones* **2022**, *6*, 147. [CrossRef]
29. Ouaddah, A.; Elkalam, A.; Ouahman, A. Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 523–533. [CrossRef]
30. Xu, R.; Chen, Y.; Blasch, E.; Chen, G. Blendcac: A blockchain-enabled decentralized capability-based access control for iots. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1027–1034.
31. Hammi, M.T.; Hammi, B.; Bellot, P.; Serhrouchni, A. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. *Comput. Secur.* **2018**, *78*, 126–142. [CrossRef]
32. Liu, H.; Han, D.; Li, D. Fabric-IoT: A blockchain-based access control system in IoT. *IEEE Access* **2020**, *8*, 18207–18218. [CrossRef]
33. Sivaselvan, N.; Bhat, V.; Rajarajan, M. Blockchain-based Scheme for Authentication and Capability-based Access Control in IoT Environment. In Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 28–31 October 2020; pp. 323–330.
34. Khalid, U.; Asim, M.; Baker, T.; Hung, P.C.; Tariq, M.A.; Rafferty, L. A decentralized lightweight blockchain-based authentication mechanism for IoT systems. *Clust. Comput.* **2020**, *23*, 2067–2087. [CrossRef]
35. Wang, W.; Huang, H.; Yin, Z.; Gadekallu, T.R.; Alazab, M.; Su, C. Smart contract token-based privacy-preserving access control system for industrial Internet of Things. *Digit. Commun. Netw.* **2023**, *9*, 337–346. [CrossRef]
36. Guo, F.; Shen, G.; Huang, Z.; Yang, Y.; Cai, M.; Wei, L. DABAC: Smart Contract-Based Spatio-Temporal Domain Access Control for the Internet of Things. *IEEE Access* **2023**, *11*, 36452–36463. [CrossRef]
37. Novo, O. Scalable access management in IoT using blockchain: A performance evaluation. *IEEE Internet Things J.* **2019**, *6*, 4694–4701. [CrossRef]

38. Qin, X.; Huang, Y.; Yang, Z.; Li, X. LBAC: A lightweight blockchain-based access control scheme for the internet of things. *Inf. Sci.* **2021**, *554*, 222–235. [[CrossRef](#)]
39. Sultan, A.; Mushtaq, M.A.; Abubakar, M. IOT security issues via blockchain: A review paper. In Proceedings of the 2019 International Conference on Blockchain Technology, Honolulu, HI, USA, 15–18 March 2019; pp. 60–65.
40. Alvi, S.T.; Uddin, M.N.; Islam, L.; Ahamed, S. A Blockchain based Cost effective Digital Voting System using SideChain and Smart Contracts. In Proceedings of the 2020 11th International Conference on Electrical and Computer Engineering (ICECE), Dhaka, Bangladesh, 17–19 December 2020; pp. 467–470. [[CrossRef](#)]
41. Uddin, M.A.; Stranieri, A.; Gondal, I.; Balasubramanian, V. Continuous patient monitoring with a patient centric agent: A block architecture. *IEEE Access* **2018**, *6*, 32700–32726. [[CrossRef](#)]
42. Alvi, S.T.; Uddin, M.N.; Islam, L. Digital Voting: A Blockchain-based E-Voting System using Biohash and Smart Contract. In Proceedings of the 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20–22 August 2020; pp. 228–233. [[CrossRef](#)]
43. Uddin, M.A.; Stranieri, A.; Gondal, I.; Balasubramanian, V. Blockchain leveraged decentralized IoT eHealth framework. *Internet Things* **2020**, *9*, 100159. [[CrossRef](#)]
44. Sultana, T.; Almogren, A.; Akbar, M.; Zuair, M.; Ullah, I.; Javaid, N. Data Sharing System Integrating Access Control Mechanism using Blockchain-Based Smart Contracts for IoT Devices. *Appl. Sci.* **2020**, *10*, 488. [[CrossRef](#)]
45. Zaidi, S.Y.A.; Shah, M.A.; Khattak, H.A.; Maple, C.; Rauf, H.T.; El-Sherbeeney, A.M.; El-Meligy, M.A. An Attribute-Based Access Control for IoT Using Blockchain and Smart Contracts. *Sustainability* **2021**, *13*, 10556. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.