electronics

MDPI

Article

Blockchain Technology for Access and Authorization Management in the Internet of Things

Maria Polychronaki [1], Dimitrios G. Kogias [1], Helen C. Leligkou [2] and Panagiotis A. Karkazis [3,*]

[1] Department of Electrical and Electronic Engineering, University of West Attica, 122 43 Athens, Greece; m.polychronaki@uniwa.gr (M.P.); dimikog@uniwa.gr (D.G.K.)
[2] Department of Industrial Design and Production Engineering, University of West Attica, 122 43 Athens, Greece; e.leligkou@uniwa.gr
[3] Department of Informatics and Computer Engineering, University of West Attica, 122 43 Athens, Greece
* Correspondence: p.karkazis@uniwa.gr

Abstract: The Internet of Things (IoT) continues to suffer from security issues, even after 20 years of technological evolution and continuing efforts. While the decentralization of the IoT seems to be a solution for improved resource management and scalability, most of the services remain centralized, exposing IoT systems to malicious attacks. As a result, this leads to functionality failures and endangers user and data integrity. Identity and Access Management (IAM) has the ability to provide defense against a great number of security threats. Additionally, blockchain is a technology which can natively support decentralization, as well as access and authorization management techniques, using the corresponding programmable logic and leveraging cryptographic mechanisms for privacy and security. Using standardized frameworks (e.g., Decentralized Identifiers and Verifiable Credentials), a blockchain-based access and authorization solution can present the basis for a uniform decentralized IAM framework for the IoT. To this end, this paper presents a proof-of-concept design and implementation of an IAM solution based on Solidity smart contracts, targeting two areas: firstly, supporting the fact that blockchain can seamlessly provide the basis for a decentralized IAM framework, while secondly (and most importantly) exploring the challenge of integrating within existing IoT systems, avoiding redesigning and redeveloping on behalf of IoT manufacturers.

Keywords: Internet of Things; blockchain; security; accessibility; authorization; identity and access management; decentralized identities; smart contracts

Citation: Polychronaki, M.; Kogias, D.G.; Leligkou, H.C.; Karkazis, P.A. Blockchain Technology for Access and Authorization Management in the Internet of Things. *Electronics* **2023**, *12*, 4606. https://doi.org/10.3390/electronics12224606

Academic Editors: Ashwin Ashok, Yue Zhang, Yinghui Zhang, Gang Han and Ming Li

Received: 29 September 2023
Revised: 28 October 2023
Accepted: 7 November 2023
Published: 10 November 2023

1. Introduction

Recently, smart technological applications have found great adoption by the consumer public, mostly in the (smart) home/city and industry automation sectors, without leaving other sectors untouched, such as the automobile and health industries. However, the Industry 4.0 revolution has presented a great number of challenges [1–3] and requirements for integration and communication with custom-needed systems. IoT-based technology combined with blockchain seems also a very promising match, as it holds the potential of integrating with almost every state-of-the-art technology at the present time (e.g., artificial intelligence, machine learning, robotics, etc.) [4–8].

Unfortunately, despite the research and development, the topic of security continues to set challenges that may compromise IoT systems. Lin et al. [5], after studying edge computation integration in IoT, presented an analysis on the importance of security as well as privacy within the context of IoT. They managed to identify six core pillars which define IoT security:

(a) Data confidentiality exclusively to authorized users;
(b) Data integrity over communications;

(c) The continuous availability of services and data on demand;
(d) The identification of authorized devices and applications as well as the authentication of incoming data to be legitimate;
(e) Data privacy and control over them exclusively for authorized users;
(f) Trust between different things, layers and applications to preserve and obey all of the above.

Inspecting the available literature regarding blockchain solutions targeting to improve IoT security, one will realise that the majority of the research aims to cover only points a, b and e, of the above pillars (data integrity, confidentiality and privacy). Some definitive examples are in [9–12] where the research is revolved around data integrity and confidentiality, while the rest of the points are left unattended. The exception lies on a relatively small number of papers, presenting a very high-level conceptualization, which, however, focus exclusively on meeting security needs on a customized way and measuring performance indicators. A common factor of these solutions is interfering with the authorization processes, as is indicated by [13–18].

On the other hand, looking at the state of the art in the blockchain industry, besides the fact that there are no market-ready authorization solutions, every other solution to be found which supports IoT security demands a level of client customization. This proves the fact that there is a lack of consistency in how security challenges are met, leading to chaotic and unstable integrations. Both the research and industry sectors can greatly benefit from uniformity, which would come from putting in use a number of standards designed for decentralized systems and authorization purposes, such as the Decentralized Identifiers (DID) and Verifiable Credential (VC) standards.

Moreover, the question asked considering all of the above is how can this combination of standards and technologies happen in order for it to be easier for the IoT industry to integrate them without needing to rebuild already-established IoT systems or requesting customized, and hence costly, solutions? While this question possibly has more than one correct answer, our objective in this paper is to propose one possible approach, through presenting a Proof-of-Concept (PoC) solution. The target of the presented PoC is to target security in practice in IoT environments and, more specifically, access and authorization management using blockchain.

Starting with explaining how the solution can be integrated into IoT systems (Section 2), the software tools used in this PoC (Section 3) follow, while Section 4 presents the integration process of the PoC with IoT. Section 5 presents the architecture and flow of the solution, while in Section 6, the PoC implementation follows with the description of the Smart Contracts' structure and logic, as well as an IoT scenario topology for putting in use the designed solution (Section 7). Lastly, in Section 8, the results of this experimental attempt are thoroughly explained. Finally, a discussion for future research is presented in Sections 9 and 10, which conclude this research paper.

2. Related Work

Related literature has been studied, in order to gain in-depth understanding and knowledge of the subject of IAM, as well as frameworks and implementations based on blockchain technology related to IoT. High complexity, the need for scalability and the diversity of tools and communication protocols which need to be used expose IoT systems to various external threats.

In this section, we focus on the topic of Identity and Access Management (IAM), which is the core framework on which most of the authentication and authorization ser-vices are based. Understanding the core concepts of IAM implementations is crucial for designing a blockchain-based access management tool for IoT-enabled devices and ser-vices. We have also studied the standardization of related frameworks, which is of great value and key importance for the scalability of systems and solutions for access and au-thorisation management. Through using standardized frameworks, the scalability of different solutions can be increased, while through defining universal interfaces, the compatibility

of different components is ensured. Lastly, related work is presented regarding existing block-chain based IAM solutions within IoT systems, and the key point where our solution is differentiated is pointed out.

2.1. Identity and Access Management—IAM

Most of the cyber-attacks which threaten an IoT system and were mentioned in the introduction can be dealt with through applying an Identity and Access Management (IAM) model.

In more practical terms, this means the creation of a roles and rules framework which is designed and adjusted to the corresponding system, in order to ensure the authentication and access of users, services and devices by demand. Integrating tools using technologies such as Public Key Infrastructure (PKI) and Certificate Authorities (CAs) is the most common solution for authentication and access management in computing environments.

In 2019, Kettani et al. [13–19], in an effort to build an access management system based on RFID, PKI and blockchain technologies, presented the four principles an IAM model should follow. The first three of these principles can be fulfilled efficiently using blockchain technology:

1. Authentication: Ensures the identity of a user or device in the context of an organization through validating their authentication credentials (username/password, fingerprint, etc.).
2. Authorization: Roles and rules policies are engaged, defining different access levels which correspond to the application's user access hierarchy.
3. Identity Management: A system responsible for registering as well as managing an entity's (user, service or device) identity within the application's environment.
4. Federated Identity Management: Third-party services which act independently of a system while providing the certification of both a user's identity and their access level to various services and platforms (Single Sign On—SSO, Open Authorization—OAuth and OpenID).

2.2. Blockchain Standards

Centralized environments enforce IAM in a system through using a centralized server as a proxy, to which entities are forwarded in order to authenticate themselves. At the same time, services also use this centralized server to validate the authorization of an entity using its unique id (usually a token). In the contrary, in a decentralized environment, all nodes of the network must be in agreement regarding the state of the shared information and do not rely on a single entity. Blockchain, being a technology that inherently operates based on a consensus algorithm between nodes, is considered to be one of the best technologies to support IAM models [19]. Thus, the shared information (commonly known as the ledger) can revolve around IAM policies and data. Consequently, some standards have been defined specifically for the purposes of security systems (whether they are an IAM implementation or different).

Decentralized Identifiers (DIDs) [20] and Verifiable Certificates (VCs) [21] are the most well-known standards for decentralized environments, developed and proposed by the World-Wide Consortium (W3C). Both of these standards are defined by their characteristics and the information they represent, as well as their functionalities in the context of a decentralized environment, regardless of whether it is considered a blockchain, a Distributed Ledger Technology (DLT) or any other. They both rely on the existence of a verifiable data registry, in which information is written and can be cryptographically verified.

A DID is defined by an alphanumeric string of characters, unique within the registry's lifecycle, which consists of three parts: the DID scheme followed by the specific implementation, the DID method through which one can verify the specific DID and the Method-specific Identifier. Moreover, each DID must be able to be connected with one DID document (available in JSON or JSON-LD form) containing all the necessary information

regarding the entity which it represents, as well as all the possible cryptographic methods that can be used for verification.

On the other hand, VCs are intertwined with the creation of DIDs for the reason that VCs can only be used to authenticate a property corresponding to an entity holding a DID in the same decentralized registry environment. VCs hold all the cryptographic material which can validate a property, with the purpose of preserving as much as possible the corresponding entity's privacy. An exceptional example of such cryptographic method is the use of Zero-Knowledge Proofs (ZKPs). For this purpose to be achieved, a VC is composed of two parts: the VC itself, which contains any sensitive (or not) information regarding a certain certificate (e.g., driver's license, etc.), and the second part, which is the verifiable presentation, a cryptographically encrypted presentation of the VC allowing a third entity to digitally verify (or prove) a claim regarding the corresponding entity.

2.3. Blockchain-Based IAM

Rayna et al. [22], targeting the analysis of the weaknesses as well as the benefits of combining blockchain and IoT, present a number of improvements which can be achieved while integrating IoT and blockchain. More specifically, both the authentication and authorization of users and devices can benefit from the decentralization feature that blockchain offers; the autonomy of the devices can also be improved when no intermediaries are involved, while the security of IoT can surely be benefited by the strong cryptographic methods used by blockchain.

Furthermore, after studying some of the experimental architectures designed exclusively for IAM with blockchain, such as the ones from [23,24], it becomes obvious that integrating blockchain-based solutions in an appropriate way can significantly increase the defence of the IoT system against some of the most common cyber-attacks, which inherently threaten IoT systems. These are the DoS or DDoS attacks which occur usually when there is a Single Point of Failure (SPoF), as well as the Link Attack, during which an attacker attempts to backtrace a user's public key, giving him/her access to personal and possibly sensitive information.

When an ecosystem is dependent on centralized services, the scalability can be exponentially decreased, contrary to a distributed or decentralized-based ecosystem. The more nodes a network consists of, the more integrity it gains, making it extremely difficult for an attacker to attack all nodes simultaneously.

In [25], Novo presents, with details, an architecture and its implementation, where access management is achieved through "Management Hubs". These essentially com-prise the blockchain network, while, at the same time, they also act as interfaces for devices' interconnection. Both in this example and in those showed in [23,24], it is proven that the transaction execution presents certain delays (of milliseconds), in contrast with a centralized implementation, due to the nature of the distributed network and the consensus algorithms which must run by all nodes. However, these delays can be improved through changing the algorithm model, depending on the implementation use case and the governance type of the corresponding blockchain network.

Further focusing on the IoT sector, for the last three years, the topic of blockchain-based access control and authorization has been given a lot of attention. In [26], the authors present a blockchain-based authorization solution for the IoT cloud, in their effort to overcome the resource limitations of existing solutions, introducing a solution to replace the classic IoT cloud framework for authorization exclusively. In [27], a double-layer blockchain solution is presented for ensuring the high network performance of ABAC-based authorization for IoT devices. However, this solution does not ensure the ability for wide integration and adoption.

Mishra et al. introduce in their work [28] a decentralized authorization model, based on blockchain technology, which showcases the usability and benefits of implementing a blockchain-based solution for IoT data sharing. Their proposed architecture takes under consideration all possible actors for an authentication management system, while it also

provides token-based generation and verification processes. Lastly, Chen et al. [29] present a fine-grained solution, Policychain, which is a blockchain-based ABAC system for the decentralized management of access and authorization policies, targeted for Industry 4.0.

However, none of the aforementioned solutions include any standardized framework that introduces adoption and integration limitations, such as the DID and VC frameworks.

3. Software Tools

Related literature [30–33] regarding the different software tools which are used for blockchain development showed that the most popular platforms for non-DeFi applications but which, at the same time, are appropriate for IoT applications, are the Ethereum Ecosystem, Hyperledger Fabric, IBM Blockchain and R3 Corda. However, the most complete framework of tools is offered by the Ethereum Ecosystem, since there are software tools and development kits for end-to-end solutions, starting from the network itself, reaching all the way to the end user's wallet.

For the access and authorization management solution presented in this paper, the following components were considered:

- Ethereum 2.0 Blockchain Platform (https://ethereum.org/en/, accessed on 27 September 2023): Offers a number of different public Ethereum networks. However, only the Mainet is the one which records and traces the crypto-coin Ether. There are also solutions which provide private networks where transaction fees are significantly decreased. It should be mentioned that any EVM-compatible platform is suitable, due to the fact that solidity and therefore smart contracts are not platform-specific but EVM-dependent.
- Solidity v.0.8.12 (https://soliditylang.org/, accessed on 27 September 2023) programming language: A Javascript-like syntax programming language, flexible and evolved to the point that it has been proven to be Turing-complete. It offers smart contract developers the means to build and implement business and programmable logic which aims to validate or change the state of the blockchain ledger. Variables and data are controlled and manipulated via smart contracts, but their full history is logged in the ledger.
- Remix IDE v.0.37.2 (https://remix-project.org/, accessed on 27 September 2023): A complete web-based and open-source development environment for writing, compiling and debugging solidity smart contracts. A desktop version of the IDE is also available.
- Metamask v.11.4.1 (https://metamask.io/, accessed on 27 September 2023): A popular, open-source wallet for the Ethereum network for managing crypto-coins and storing the cryptographic keys of an Ethereum account.
- DID Standard v1.0 (https://www.w3.org/TR/vc-data-model/, accessed on 27 September 2023): A DID is defined by an alphanumeric string of characters, unique within the registry's lifecycle, which consists of three parts: the DID scheme followed by the specific implementation, the DID method through which one can verify the specific DID and the method-specific identifier.

4. Integration with IoT

The solution presented in this paper is created in the context of being easily integrated with IoT systems. This is achieved through providing smart contract interfaces for managing authorization and access rights for "things". In favour of presenting the following solution, we assume an IoT system based on a five-layer architecture [34], as seeing in Figure 1 [34]. The ability for internet communication is vital for the functionality of this solution, since the interaction with a blockchain network is based on it. However, bearing in mind that most IoT systems, regardless of the heterogeneity of the protocols used, provide Internet connectivity at some point, the risk of integration failure is considered low for this particular reason.

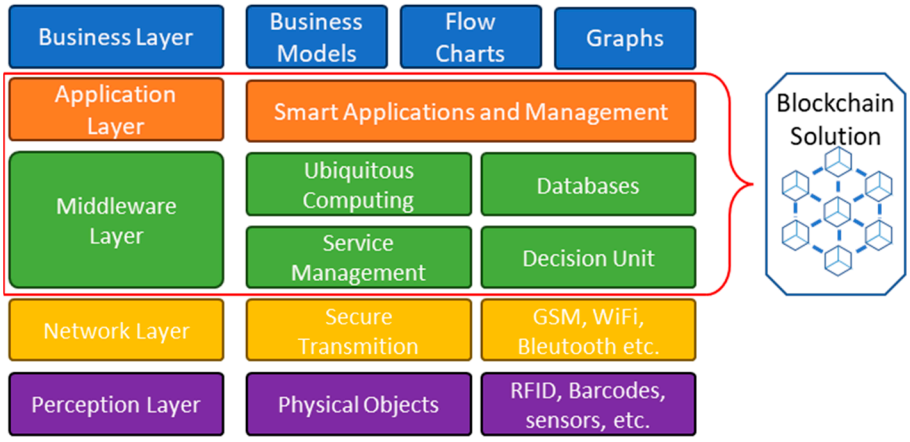

Figure 1. Access and authorization management solution in a 5-layer IoT architecture.

One crucial factor for the design process of the presented blockchain-based access and authorization management solution is the need to reach the Internet. Therefore, it should be integrated at least within the application and middleware layers (see Figure 1), where it is more likely that Internet connectivity will be available (Figure 1). However, if it is possible for all the components of the network and perception layers to access the Internet, either directly or using a secure proxy component (e.g., a gateway for extremely low-end sensors), the blockchain-based solution can be integrated with them as well.

Moreover, this referenced architecture is able to offer control regarding the authorization of both devices and services, thus giving the provider or stakeholder of applications the possibility to also integrate it with the business layer. One such way would be through offering different access and authorization levels based on a subscription plan (e.g., number of controlled devices, multiple user roles or custom-made roles).

The main goal of our design is to deliver the basis of a set of blockchain interfaces for converting traditional server–client IoT applications to partially decentralized applications. The term "partially" is used to define the fact that applications can integrate it in the back end (Figure 2) through adding only the functionalities which manage access and authorization services via the blockchain network. This results in applications modified in such a way that the end users will be already familiar with the user interface (UI), but, at the same time, the IoT environment will be enhanced with blockchain functionalities controlling access and authorization policies for all entities in the system.

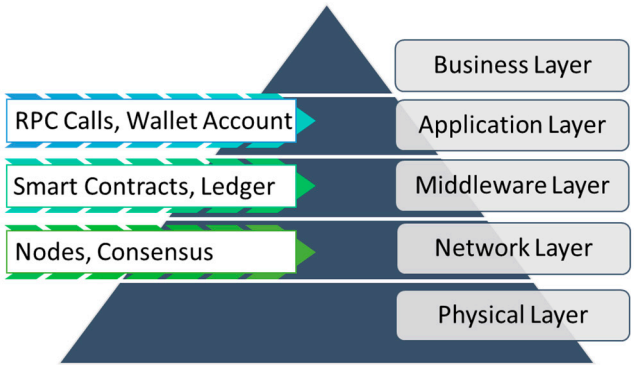

Figure 2. Partial integration of blockchain in IoT.

Figure 2 explains this kind of integration, where each high-level blockchain component is mapped to a respective IoT layer in the five-layer architecture. It is important to note that, in contrast with the solutions that can be found in recent or older literature, blockchain is not considered as one separated component, but its main sub-components are integrated in IoT.

Starting from the bottom, the blockchain network can be part of the network layer of the IoT, especially if common infrastructure is chosen (same hardware hosting both blockchain nodes and IoT servers). Moving one layer up, the smart contracts and the ledger are integrated within the middleware layer as this is where communication services and databases are. IoT services can also communicate with the smart contracts and the network in order to retrieve or add ledger information. Finally, the IoT applications can integrate Web3 technology in order to acquire the ability to make RPC calls towards the blockchain network and host the user's account (public and private key).

5. Implementation Context, Components Structure and Flow Diagram

Let us consider a simple IoT system where smart home automations take place in a workplace environment. For example, people working in an office room have several smart devices (e.g., smart plugs, light switches and an air conditioner) and smart hubs (e.g., SmartThings hub, Philips, Google Home and Amazon) in order to control environmental conditions within the room. Using smartphone applications, people can easily set and delete rules regarding automation for controlling light, room temperature and humidity. Our PoC is designed in such a way that manufacturers can integrate the blockchain functionalities of our smart contracts in their application, making it Web 3.0-compatible, seamlessly for the end users. Ideally, the user experience (UX) of manufacturers' applications should not be altered at all, whereas the functions behind the scenes of the graphical interface can create and send transactions when the user performs an action that should be recorded in the common ledger anonymously (e.g., adding a new device in their smart hub or creating a new automation rule).

In this section, the components of an authorization system based on blockchain technology are presented and described, designed in such a way targeting the seamless integration with existing IoT systems and aiming at verifying user and device certificates. At the same time, the proposed solution integrates standardized frameworks, in order to achieve consistency and the capability for wider-scale adoption.

Assuming the existence of an IoT system that takes advantage of blockchain for access and authorization, we recognize at least four different entities within it:

A. Users: These are the end users who interact with the system, usually via a mobile application, and either send through it a command or receive feedback (an alert) for an event.
B. Gateways: Devices placed mostly in the middleware layer, acting as an intermediary for other devices which cannot perform certain functions (e.g., communicate via the Internet).
C. Devices: Any device acting like a sensor or actuator and participating in the system through either providing or consuming data. Many low-end devices cannot reach the Internet, in which case, they use a gateway for performing their functionalities (e.g., sending data to the cloud or communicating with the blockchain network).
D. Blockchain: A decentralized network of nodes which run smart contracts in order to share a distributed and immutable ledger. All data concerning the authorization and access management (tools, policies and keys) are logged onto the ledger and the information can be verified by any of the network nodes.

Additionally, Figure 3 depicts a flow diagram with the different roles/components which exist in our PoC scenario, as well as the ways with which they can interact with each other. Moreover, the steps which components can and must complete are numbered. These steps form a guideline for creating Decentralized Identifiers (DIDs) [20] for all system components (users, gateways and devices), but also any verifications needed for each one of them. A DID is defined by an alphanumeric string of characters, unique within the registry's lifecycle, which consists of three parts: (i) the DID scheme followed by the specific implementation, (ii) the DID method through which one can verify the specific DID and (iii) the method-specific identifier.

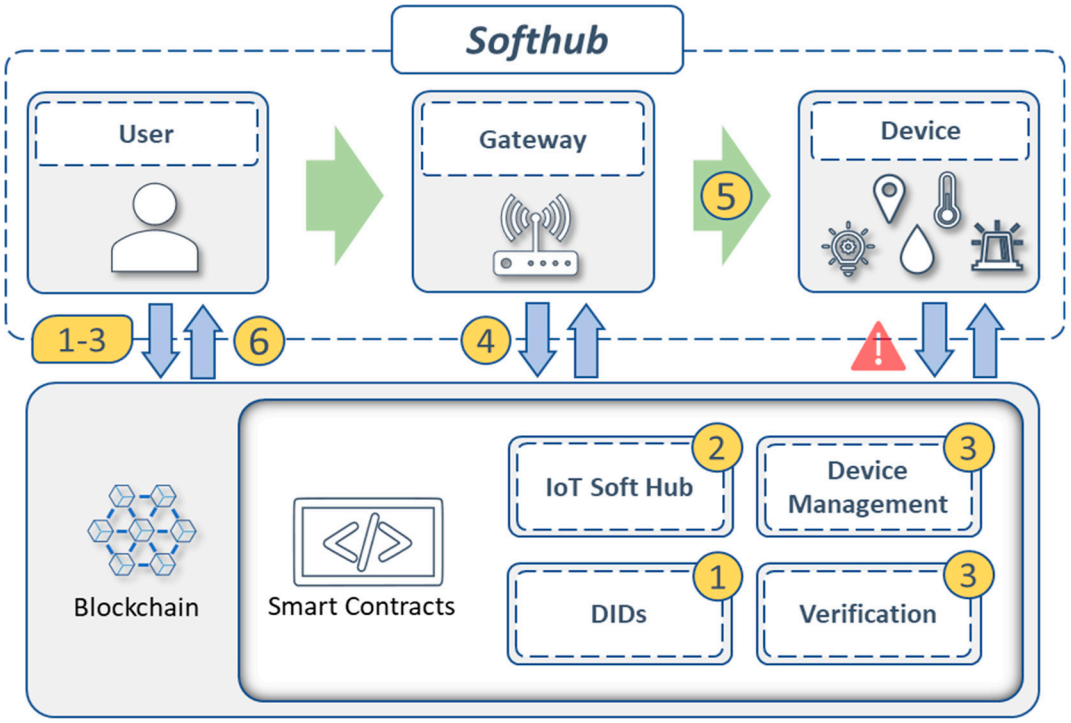

Figure 3. Access and authorization management architecture and flow diagram.

Moreover, each DID must be able to be connected with one DID document (available in JSON or JSON-LD form) containing all the necessary information regarding the entity which it represents, as well as all the possible cryptographic methods that can be used for verification. DIDs play a crucial role in the broader movement toward self-sovereign identity, where individual entities have greater control and ownership of their digital identities, reducing reliance on centralized identity providers and enhancing security and privacy [35–37].

In this implementation, the creation of a "softhub" is mandatory, which acts as a digital equivalent of the physical IoT system. Softhub is a term formed specifically to describe the need for a digital equivalent of an IoT system, in the context of the solution described in this paper. In other words, when a physical device is part of an IoT system, the blockchain network will log that the device's DID is bound in the corresponding softhub, also characterised by a DID.

The following further describes the steps shown in Figure 3:

1. Users create for themselves a unique DID, by executing the corresponding smart contract (DIDs) using their account.
2. Users create or join a softhub, by executing the corresponding smart contract (IoT SoftHub) using their account.
3. Gateways and devices also acquire a DID, by executing the same smart contract as step 1 while softhub administrators claim them by executing the Device Management smart contract.
4. Attention should be given if devices do not have the capability of directly communicating with the Blockchain. If there are gateways representing limited functionality devices, they repeat steps 1–3, on behalf of them.
5. Every time a process needs to be completed by any component of the softhub, the permission to do so must be verified through querying the ledger via the verification smart contract.
6. Finally, when necessary, the execution of certain smart contract functions can provide feedback to the user via events, informing them of the new status of the softhub

(e.g., a device's ownership changed, or an unauthorized device attempted to communicate within the softhub).

By the end of this experiment, it is expected that three different kinds of proofs are to be provided for each entity, which will be demonstrated in the Section 8:

1. Proof of membership in a particular IoT system (softhub);
2. Proof of authorization management rights over the IoT system (mostly concerns users rather than devices);
3. Proof of control over a device.

6. Programmable Logic—Smart Contracts

In this section, the logic behind the smart contract functionalities is described in detail. These are divided in four smart contracts based on the context in which they operate and the state which they manipulate. Two smart contracts function as factories, meaning that they hold state variables and create instances, while the other two smart contracts act as services, containing only functions for manipulating the state and providing authorization services. Each smart contract is characterized by its variables' definition and how their content changes based on the contract's functions, while everything is recorded as transactions.

Let us keep in mind that one of the key factors of authentication frameworks is the logging capability of any process. Even administrative processes are recorded and can be available to users only at the top layer of the security hierarchy. Thus, in this particular PoC, the ledger contains two crucial pieces of information supporting access and authorization:

- The current state of the authorization policies;
- The history of the variables' content, or else the logging of all processes altering the authorization policies.

The pseudocode for all four smart contracts described below is given in Appendix A of this paper.

6.1. DidFactory Smart Contract

The first smart contract is responsible for creating unique Decentralized Identifiers—DIDs. Each and every entity within the IoT system using this solution must be represented by a personal DID which corresponds strictly to one and only one Ethereum account. Table 1 shows all the variables which the DidFactory smart contract uses for that end, following the W3C standard. These variables aim to keep a record of correspondence between the Ethereum accounts, softhubs, devices and the DIDs, while also checking if an account has already registered or not.

Table 1. Variables used by the DidFactory smart contract.

Variables—DidFactory.sol			
Name	**Type**	**Description**	**Input Fields**
Did	struct	Holds the did of the corresponding entity.	scheme, method, path, property, registrationNo
dids	array	Lists all the DIDs created with this smart contract.	none
identityToAccount	mapping	Maps the did object to the owner's ethereum account.	None
accountIsRegistered	mapping	Flags as true every account which gets a Did.	None

Regarding functionality, this smart contract has a constructor which initializes the first DID string, which belongs to the smart contract itself, while there is one more function for creating DIDs on call, which takes a single parameter as an input and that is the property of the entity being registered (e.g., administrator, user, softhub, device, etc.). The pseudocode of this smart contract is given in Table A1.

6.2. SofthubFactory Smart Contract

Similarly, with the DidFactory smart contract, SofthubFactory contains all the necessary variables for the creation as well as the logging of all the softhubs. As stated earlier, the logic of this solution to map each real IoT system to one softhub. Table 2 lists all the variables used by the SofthubFactory smart contract. These aim to map each softhub with every device registered and one owner. Also, they map each device with one and only one serial number and create and maintain a record of softhub history for every device.

Table 2. Variables used by the SofthubFactory smart contract.

Variables—SofthubFactory.sol			
Name	**Type**	**Description**	**Input Fields**
Device	struct	creates a Device object corresponding to a physical device	deviceID, hubID, deviceType
Softhub	struct	creates a Softhub object corresponding to a physical IoT system	DID, ownerDID, name
deviceHubHistory	mapping	maps each Device with an array of softhubs which it was claimed from in the past	none
OwnerToHub	mapping	maps each Softhub to its owner's Ethereum account	none
rnToHub	mapping	maps each device to its unique serial number	none
devToHub	mapping	maps each device to a softhub (if claimed)	none

Regarding functionality, this smart contract has one constructor and two different functions for creating softhubs and registering devices. The constructor creates the first softhub, whose administrator is the smart contract itself, which exists for hosting any device after being registered and before being claimed by any user. This way, the devices' history is always being recorded, whether it is claimed by a softhub or not.

Moreover, there is also a function for creating a softhub through an external call by any administrator, while the commands within the function manipulate the first three variables of Table 2 accordingly. The second function also changes the remaining variables, with the purpose of registering a new device in the ledger. Two input parameters are needed for device registration, which are the device's type (e.g., temperature_sensor, smode_detector, etc.) and the device's registration number. The reason for using and inserting a parameter such as the registration number is to offer a way for the end user to quickly query and insert a device using a QR code containing its unique product number. The pseudocode of this smart contract is given in Table A2.

6.3. DeviceManagement Smart Contract

The third smart contract is responsible for all operations regarding device management. Through using the inheritance capability of the solidity language, the DeviceManagement smart contract can access and manipulate the state of every variable mentioned in the

previous smart contracts. Users can invoke four different functions in order to claim a device, remove one from a softhub or transfer it to another.

While claiming and removing a device requires the invocation of only one function, the process to transfer one from a softhub to another requires two functions. For enhancing the integrity of the transferability, the process itself is divided into two phases. The first part is for ensuring that the device is to be transferred to a specific softhub, through using the Ethereum account of the next owner (claimer) to bind the device. After the first owner executes this transaction, the second phase follows, where the next owner must claim it using the special function made for this purpose ("claimTransferedDevice"). This way, the account claiming a transferred device is ensured to be the one which the device is already bound to.

These four functions can be invoked only externally, meaning that other contracts cannot invoke them, not even contracts inheriting from the DeviceManagement smart contract. This is a definitive example of how solidity language provides programmers with native tools for low-level security, while the code design should be carefully examined to prevent unwanted exploitation or exposed security gaps. The pseudocode of this smart contract is given in Table A3.

6.4. Verification Smart Contract

The fourth and final smart contract of the proposed solution is the one used for verification between entities. In contrast with the DeviceManagement smart contract, the Verification contract, while not initializing new variables, only reads the already existing variables from the DidFactory and SofthubFactory contracts. There are four public and external functions within this smart contract which are used for providing the callers with the information, whether an entity is allowed or not, for an action. The functions require two mandatory inputs (the DIDs of two entities), and they return a Boolean value for these four statements:

- The device with the DID [_deviceDID] is currently claimed by (and thus belongs to) the softhub whose administrator is represented by the Ethereum address [_address].
- The softhub with the DID [_hubDID] is bound (thus can only be controlled) by the user represented by the Ethereum address [_address].
- The device with the DID [_deviceDID] is part (and thus can interact with) of the softhub with the DID [_hubDID].
- The Ethereum account which just executed a function is an administrator of the softhub with the DID [_hubDID].

These four functions can be called by any entity within an IoT system, in order to get confirmation that another entity has indeed the permission and authorization to perform an action (e.g., user access to data; configuration changes by either users, services or devices; etc.). The importance of this smart contract is that an entity is able to confirm access and authorization for other entities, but only using their DIDs or users' public Ethereum accounts. Both DIDs and the Ethereum accounts are information which are considered public and do not expose any information regarding the characteristics of the corresponding entities. The pseudocode of this smart contract is given in Table A4.

7. Scenario Topology

In order to extract results, a topology (Figure 4) was designed to test the functionalities of the smart contracts. We have assumed the existence of three simple IoT systems, which are represented by three softhubs with the names "Softhub1", "Softhub2" and "Softhub3", respectively. Each one of these can be controlled by one and only one user, who is considered the administrator of the corresponding softhub. It is also assumed that there are two different sensor devices (Device 1 and Device 2) which have access to the Internet and are fully qualified to communicate directly with the blockchain, having their own Ethereum accounts.

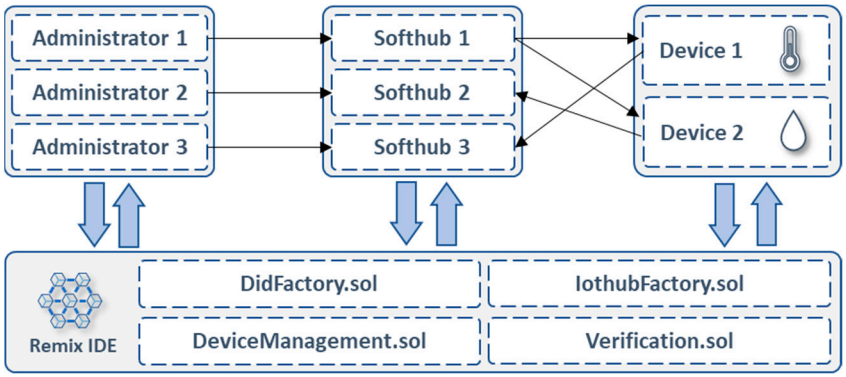

Figure 4. Testing softhub topology.

The procedure of the experiment aims at forming the topology described and shown in Figure 4 through executing a series of transactions invoking all functions from the first three smart contracts of the access and authorization management PoC. After those transactions, the fourth smart contract will be invoked in order to verify the state the access and authorization rules.

During the execution of the experiment, the state of the ledger was recorded, and changes before and after each transaction were noted. The exact transactions which were executed, after the deployment of the smart contracts in the Remix IDE using the first available account, are:

1. Select the second account available by Remix IDE (0xAb8483F64d9C6d1EcF9b849Ae-677dD3315835cb2) and create a personal user DID with the property of "Administrator".
2. Using the same account, create a softhub with the name "testhub1".
3. Regardless of the account selected, register (create DIDs for) two devices:
   i. One temperature sensor device with the serial number "abc1234"
   ii. One smoke detector device with the serial number "def5678"
4. With the second account selected, claim both registered devices into "testhub1".
5. Create two more personal user DIDs with the property of the "Administrator" using the next two (third and fourth) available Ethereum accounts of Remix IDE:
   i. 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
   ii. 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
6. Repeat step two, to create two more softhubs corresponding to the administrators of step 5:
   i. "testhub2"
   ii. "testhub3"
7. With the second account selected, execute phase 1 for transferring Device1 (serial number: abc1234) from softhub1 to softhub2.
8. With the fourth account selected, make one attempt to claim Device1 to softhub3, while the device is bound to softhub2 (transaction should fail).
9. With the third account selected, complete the transfer of Device1 and claim it to softhub2.
10. With the first account selected, remove Device2 (serial number: def5678) from softhub2.
11. With the fourth account selected, claim Device2 (serial number: def5678) to softhub3.
12. Verify that Device1 (serial number: abc1234) belongs to softhub1.
13. Verify that Device1 (serial number: abc1234) belongs to softhub2.
14. Verify that Device2 (serial number: def5678) belongs to softhub1.
15. Show softhub history of Device2 (serial number: def5678).
16. Verify that Device1 (serial number: abc1234) can be controlled by the account address which corresponds to softhub1.

17. Verify that Device1 (serial number: abc1234) can be controlled by the account address which corresponds to softhub2.
18. Verify that softhub1 can be controlled by the account address which corresponds to softhub2.
19. Verify that softhub1 can be controlled by the account address which corresponds to softhub3.

8. Results

After deploying the four smart contracts within the Remix environment, the transactions from the list above were executed, with the intention of observing the response after each one and the way the ledger state is formed. The console provided by Remix (version number: v0.37.1) offers the opportunity to observe exactly what happens when a transaction is minted by an Ethereum node. A transaction response is very similar to an HTTP response since it also has headers which indicate certain metadata.

For example, Table 3 shows all the execution metadata for transaction 1: create a personal user DID with the property of "Administrator". The "status" field indicates whether the transaction is succeeded or not after running and execution, while from the fields "to" and "decoded input", we can understand how the ledger's state variables are formed.

Table 3. Execution of transaction 1 (create a personal user DID with the property of "Administrator").

Status	**Transaction Mined and Execution Succeed**
transaction hash	0xb6d05334b6082. . .23142afb7daf45
from	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to	DeviceManagement.createdid (string)
gas	309545
transaction cost	269169
execution cost	269169
input	0xa83. . .00000
decoded input	{"string _property": "administrator"
decoded output	{}
logs	[]
val	0 wei

After the execution of transactions 1 through 11, the state variables have been formed in a way which reflects the topology of Figure 3. Altogether, there are eight entities (three administrators, three softhubs and two devices) which all have a unique personal DID, while the administrators also are mapped with their Ethereum account. It is worth noting that transaction 8, which has a purpose of testing the integrity of the smart contract design, indeed fails. This transaction is made by an unauthorized user who attempts to claim a device after phase one of its transfer has been completed and before the authorized and bound administrator claims it.

The response in the failed transaction has a status message of "**false Transaction mined but execution failed**", while below the response metadata, the message **"transact to verification.claimTransferedDevice errored: VM error *revert. The transaction has been reverted to the initial state*"** appears. This message states that whatever code was executed by the smart contract function which was not approved by the node has been reverted and no changes in the ledger state are recorded. Nonetheless, the action of a failed transaction is also recorded in the network logs.

The final state of the ledger, as shown in Table 4, contains all the DIDs of the eight entities registered, as well as the mappings for Ethereum accounts corresponding to administrators, the administrators to each softhub, the devices' registration numbers and which softhub they are claimed to.

Table 4. Final state of ledger forming Figure 4 topology.

Data Type	Data Value
Struct DID	did:. . ./administrator91101
Struct DID	did:. . ./administrator98868
Struct DID	did:. . ./administrator54989
Struct DID	did:. . ./softhub53703
Struct DID	did:. . ./softhub37899
Struct DID	did:. . ./softhub92253
Struct DID	did. . ./device89873
Struct DID	did:. . ./device47235
Struct Softhub	Testhub1, did:. . ./softhub53703
Struct Softhub	Testhub2, did:. . ./softhub37899
Struct Softhub	Testhub3, did:. . ./softhub92253
Struct Device	temp_sensor, abc1234
Struct Device	Smoke_detect, def5678
Mapping identityToAccount	0xAb84. . .35cb2 => did:. . ./administrator91101 0x4B20. . .C02db => did:. . ./administrator98868 0x7873. . .cabaB => did:. . ./administrator54989
Mapping OwnerToHub	did:. . ./administrator91101 => Testhub1 did:. . ./administrator98868 => Testhub2 did:. . ./administrator54989 => Testhub3
Mapping rnToDev	abc1234 => did:. . ./device89873 def5678 => did:. . ./device47235
Mapping DevToHub	did:. . ./device89873 => Testhub2 did:. . ./device47235 => Testhub3

Lastly, transactions 12 through 19 aim to prove the verification functionality of the smart contracts. These eight transactions are not recorded to the ledger, due to the fact that they do not provide any kind of changes to the state of the variables mentioned. They rather provide confirmation (or disapproval) regarding the four statements of the Verification smart contract. The response of these transactions is simply a true or false statement, responding to the verification query. Table 5 shows the response for transactions 12–19 as well as the input parameters of the query made each time. The results truthfully verify the ledger state through returning a Boolean value, allowing for further programming logic to be developed based on it regarding access permissions.

Table 5. Verification transactions (12–19).

Verification Function	Parameter 1	Parameter 2	Response
verifyDeviceToSofthub	did:. . ./softhub53703	. . ./device89873	false
verifyDeviceToSofthub	did:. . ./softhub37899	. . ./device89873	true
verifyDeviceToSofthub	did:. . ./softhub53703	. . ./device47235	false
verifyDeviceOwnership	did:. . ./device89873	0xAb8483. . .835cb2	false
verifyDeviceOwnership	did:. . ./device89873	0x4B2099. . .2C02db	true
verifySofthubOwnership	did:. . ./softhub53703	0x4B2099. . .2C02db	false
verifySofthubOwnership	did:. . ./softhub53703	0x78731D. . .5cabaB	false

Observing the implementation of this PoC, it is evident that a blockchain ledger can be used as a policy keeper in order to provide a trustless but transparent source of access and authorization information and verification. The standard frameworks of DIDs (and extensively VCs) protect private information from leaking while, at the same time, uniquely characterizing all entities which participate in an IoT hub, the equivalent of an IoT system in real life. Devices, users and services acquire a DID and enrol in the ledger policy, while all communication processes can add one more step of verification throughout the whole IoT system. Thus, the components communicating can all verify each other and choose whether to proceed or not with their execution processes.

9. Discussion and Future Research

Using a blockchain-based solution for enhancing IAM functionality offers a variety of advantages regarding the security of an IoT system. Depending on the architecture and solidity techniques used, blockchain is able to counteract some of the most common attacks threatening IoT. Indicatively, some of these are as follows:

- Enhancing security mechanisms for securely updating hardware firmware;
- Device and access management on larger-scale networks where there is a large number of identical devices (e.g., smart cities);
- Add one more authorization layer on off-chain data accessibility;
- Design more secure high-level applications, especially when devices cannot be fully protected and secured;
- Counteract the device impersonating attack using unique DIDs in combination with blockchain's public key cryptography.

One of the research topics which can be further developed is the possibility of using solidity's Events functionality, in order to build a multi-factor authentication process for applications' accessibility. Events are a solidity function which is able to send feedback from the smart contract to the front-end decentralized application, upon triggering within the code. Thus, it is possible to design a group of smart contract functions which allow multi-factor authentication.

Moreover, in order to enhance even more the privacy of the presented solution, ZKP cryptography can be used for the proper implementation and integration of the Verification Credentials (VCs). In such a case, the Verification smart contract can be replaced by a custom-made and softhub-targeted smart contract, which could be designed to verify specific ZKPs, instead of having simple solidity functions querying the ledger with human-readable information. ZKPs will represent the access or authorization rights of an entity, which, as an information, piece is not human-readable. This way, the solution can be scaled and extend its capabilities regarding access and authorization management.

Moreover, the performance of this solution should not be neglected in any way. Currently, the PoC described in this paper is not deployed in any public mainet or testnet. The deployment of such a solution should be tested in different blockchain networks where the consensus and communication of its nodes is differentiated. Performance can be tested in terms of transaction rate as well as the network's defence against malicious nodes, depending on the number of nodes and the type of consensus they use in order to ensure the state of the ledger and whether it complies with the smart contracts' logic.

Finally, although it can be considered resource-costly, using one top-layer smart contract for the users, the four smart contracts presented above could be programmed and tailored to the end users' needs. This way, customizations of the user roles, the softhub properties and accessibility levels would be defined as an end user wishes. This would result in an exponential increase in smart contracts which a network must manage and run, therefore affecting network and system performance. However, it would also offer end users hardcoded privacy and blockchain control, meaning that the variables and functions mentioned in the DIDFactory and SofthubFactory contracts would be visible, controlled and invoked exclusively by each corresponding user only.

10. Conclusions

In this paper, a blockchain-based access and authorization PoC was presented. The main purpose of our research was to examine whether a blockchain-based authorization system can be designed in such a way, providing easy integration with already existing IoT environments. Our hypothesis was that, similarly to the oAuth framework, if certain standards are used in order to build a uniform framework for blockchain-based IAM in the IoT, then integration in IoT systems will be seamless. Those standards include DIDs and VCs. The PoC presented in this paper is the first step intending to verify this hypothesis. Its architecture is designed based on the fact that the ledger's state can be utilized as an IAM policy keeper, as well as its history as an immutable and undeletable logging system. At

the same time, solidity functions can be used as programmable interfaces for accessing and authorizing services. This way, the blockchain can be used as a decentralized Certificate Authority, while smart contracts contain custom policies, defined at the end-user level.

Blockchain can make a positive contribution to enhancing the security of not only IoT but internet applications and services. It is possible to design a blockchain solution in such a way that it can be integrated into already-operating systems. The way to achieve this, lies in the adoption of a specific logic when designing the smart contracts. This logic can be compared to that of APIs, with the difference that instead of giving remote access to data and resources, access to limited and specific solidity executable code is given, thus forming a stable and concrete access and authorization system, which is built using cryptographic mechanisms. Needless to say, blockchain is currently the most suited technology to natively integrate them.

In conclusion, a blockchain-based IAM solution is able to offer a great variety of benefits in IoT systems, especially given the latest tends toward decentralization using edge computing. Of course, the biggest question which arises after this research is whether blockchain has the potential of being the basis for a new cutting-edge decentralized and universal IAM framework standard, such as the OAuth or OpenID solutions that exist currently and are universally used over the digital industry and Internet.

Author Contributions: Conceptualization and methodology, M.P.; writing—original draft preparation, M.P.; writing—review and editing, M.P.; supervision, D.G.K., H.C.L. and P.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: Our work is open in Gitlab and can be accessed via this link: https://gitlab.com/maria204/blockchain-based-iam-for-iot.git, accessed on 27 September 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this appendix, the pseudocode of the four smart contracts explained in Section 5 is shown.

Table A1. DID Factory smart contract pseudocode.

```
DID FACTORY SOL
INIT MAPPING identityToAccount, accountIsRegistered
SET nonce <- 0
SET identifierLenth <- 5
SET modulus <- CALCULATE power (10, identifierLength)
INIT STRUCT did (scheme, method, path, property, registrationNo, did)
CONSTRUCTOR
        SET accountIsRegistered[contract account] <- TRUE
        CALL createDid("contract")
END CONSTRUCTOR
FUNCTION createDid (INPUT property)
        IF account of sender IS registered THEN
                break transaction
        ELSE
                SET accountIsRegistered[sender account] <- TRUE
                SET random <- generateRandomDigits
                SET didString <- CONCATENATE("did:","ethblockchain:","/application/", property, random)
                SET did <- CREATE STRUCT INSTANCE did
                SET identityToAccount[account of sender] <- did
END FUNCTION
FUNCTION generateRandomDigits
        SET input <- CONCATENATE (block.timestamp, msg.sender, nonce)
        SET castUint <- CALCULATE KECCAK256 (input)
        INCREASE nonce
        OUTPUT castUint % modulus
END FUNCTION
```

Table A2. Softhub Factory smart contract pseudocode.

```
SOFTHUB FACTORY SOL
INIT STRUCT device(deviceID, hubID, deviceType), softhub (did, ownerDid, friendlyName)
INIT MAPPING deviceHubHistory, ownerToHub, rnToDev, devToHub, hubDeviceCount,
didToSofthub, didToDevice
CONSTRUCTOR
    IF msg.sender did property IS NOT "contract"
        break transaction
    ELSE
        SET contractSofthubDid <- CALL createDid ("softhub") FROM didfactorycontract
        SET contractSofthub <- CREATE STRUCT INSTANCE softhub
        SET ownerToHub[contract account] <- contractSofthub
        didToSofthub[contractSofthubDid] <- contractSofthub
        SET hubDeviceCount[contractSofthub] <- 0
END CONSTRUCTOR
FUNCTION createSofthub (INPUT name)
    IF sender IS NOT registered as administrator THEN
        break transaction
    ELSE
        SET softhubdid <- CALL createDid ("softhub") FROM didfactorycontract
        SET softhub <- CREATE STRUCT INSTANCE softhub
        SET didToSofthub[softhubdid] <- softhub
        SET ownerToHub[msg.sender] <- softhub
        SET hubDeviceCount[softhubdid] <- 0
END FUNCTION
FUNCTION registerDevices (INPUT type, deviceRN, softhub)
    SET deviceDid <- CALL createDid ("device") FROM didfactorycontract
    SET device <- CREATE STRUCT INSTANCE device
    SET didToDevice[deviceDid] <- device
    SET rnToDev[deviceRN] <- device
    PUSH deviceHubHistory[device] <- 0
    INCREASE hubDeviceCount[contractSofthub]
END FUNCTION
```

Table A3. Device Management smart contract pseudocode.

```
DEVICE MANAGEMENT SOL
FUNCTION claimDevice (INPUT deviceRN)
    IF the hubId property of the device IS NOT 0
    OR
    IF the device's hub IS NOT 0
    THEN
        break transaction
    ELSE
        SET rnToDev[_deviceRN].hubID <- identityToAccount[msg.sender];
        SET devToHub[rnToDev[_deviceRN]] <- ownerToHub[msg.sender];
        INCREASE hubDeviceCount[ownerToHub[msg.sender]]
        PUSH deviceHubHistory[rnToDev[_deviceRN]] <- ownerToHub[msg.sender]
END FUNCTION
FUNCTION changeHub (INPUT deviceRN, targetOwner)
    IF the device's hub IS NOT the hub of the msg.sender THEN
        break transaction
    ELSE
        SET rnToDev[_deviceRN].hubID <- identityToAccount[_targetOwner]
        SET devToHub[rnToDev[_deviceRN]] <- 0
        DECREASE hubDeviceCount[ownerToHub[msg.sender]]
END FUNCTION
FUNCTION claimTransferedDevice (deviceRN)
```

```
        IF the binded address of the device IS NOT the same as the msg.sender THEN
            SET rnToDev[_deviceRN].hubID <- identityToAccount[msg.sender]
            SET devToHub[rnToDev[_deviceRN]] <- ownerToHub[msg.sender];
            INCREASE hubDeviceCount[ownerToHub[msg.sender]]
            deviceHubHistory[rnToDev[_deviceRN]] <- ownerToHub[msg.sender]
END FUNCTION
FUNCTION removeFromHub (deviceRN)
        IF device DOES NOT belong to the hub of the sender
            break transaction
        ELSE
            SET rnToDev[_deviceRN].hubID <- 0
            SET devToHub[rnToDev[_deviceRN]] <- 0
            DECREASE hubDeviceCount[ownerToHub[msg.sender]]
            PUSH deviceHubHistory[rnToDev[_deviceRN]] <- 0
END FUNCTION
```

Table A4. Verification smart contract pseudocode.

```
VERIFICATION SOL
FUMCTION verifyDeviceOwnership (INPUT deviceDID, address)
        IF devToHub[didToDevice[deviceDID]] I == ownerToHub[address]
            OUTPUT TRUE
          ELSE
            OUTPUT FALSE
END FUNCTION
FUMCTION verifySofthubOwnership (INPUT hubDID, address)
        IF ownerToHub[address] == didToSofthub[hubDID]
            OUTPUT TRUE
          ELSE
            OUTPUT FALSE
END FUNCTION
FUMCTION verifyDeviceToSofthub (INPUT softhubDID, deviceDID)
        IF devToHub[didToDevice[deviceDID]] == didToSofthub[softhubDID]
            OUTPUT TRUE
          ELSE
            OUTPUT FALSE
END FUNCTION
FUMCTION verifyAdminToSofthub (INPUT softhubDID, address)
        IF address == didToSofthub[softhubDID]
            IF identityToAccount[address].property IS "administrator"
                OUTPUT TRUE
            ELSE
                OUTPUT FALSE
          ELSE
            OUTPUT FALSE
END FUNCTION
```

References

1. Jepsen, S.C.; Mork, T.I.; Hviid, J.; Worm, T. A Pilot Study of Industry 4.0 Asset Interoperability Challenges in an Industry 4.0 Laboratory. In Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 14–17 December 2020; pp. 571–575. [CrossRef]
2. Rikalovic, A.; Suzic, N.; Bajic, B.; Piuri, V. Industry 4.0 Implementation Challenges and Opportunities: A Technological Perspective. *IEEE Syst. J.* **2021**, *16*, 2797–2810. [CrossRef]
3. Zalozhnev, A.Y.; Ginz, V.N. Industry 4.0: Underlying Technologies. Industry 5.0: Human-Computer Interaction as a Tech Bridge from Industry 4.0 to Industry 5.0. In Proceedings of the 2023 9th International Conference on Web Research (ICWR), Tehran, Iran, 3–4 May 2023; pp. 232–236. [CrossRef]
4. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
5. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. [CrossRef]

6. Vashi, S.; Ram, J.; Modi, J.; Verma, S.; Prakash, C. Internet of Things (IoT) A Vision, Architectural Elements, and Security Issues. In Proceedings of the International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), Palladam, India, 10–11 February 2017. [CrossRef]
7. Granjal, J.; Monteiro, E.; Silva, J.S. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1294–1312. [CrossRef]
8. Atlam, H.; Alenezi, A.; Alassafi, M.; Alshdadi, A.; Wills, G. Security, Cybercrime and Digital Forensics for IoT. In *Intelligent Systems Reference Library*; pp. 551–577. Available online: https://link.springer.com/chapter/10.1007/978-3-030-33596-0_22 (accessed on 27 September 2023).
9. Wu, W.; Liu, E.; Gong, X.; Wang, R. Blockchain Based Zero-Knowledge Proof of Location in IoT. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7. [CrossRef]
10. Chuang, I.-H.; Guo, B.-J.; Tsai, J.-S.; Kuo, Y.-H. Multi-graph Zero-knowledge-based authentication system in Internet of Things. In Proceedings of the ICC 2017-2017 IEEE International Conference on Communications, Paris, France, 21–25 May 2017; pp. 1–6. [CrossRef]
11. Muthamilselvan, S.; Praveen, N.; Suresh, S.; Sanjana, V. E-DOC Wallet Using Blockchain. In Proceedings of the 2018 3rd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 15–16 October 2018; pp. 989–993. [CrossRef]
12. Naik, N.; Jenkins, P. Self-Sovereign Identity Specifications: Govern Your Identity Through Your Digital Wallet using Blockchain Technology. In Proceedings of the 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Oxford, UK, 3–6 August 2020; pp. 90–95. [CrossRef]
13. Carnley, P.R.; Kettani, H. Identity and Access Management for the Internet of Things. *Int. J. Futur. Comput. Commun.* **2019**, *8*, 129–133. [CrossRef]
14. Vallois, V.; Mehaoua, A.; Amziani, M. Blockchain-based Identity and Access Management in Industrial IoT Systems. In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 623–627.
15. Wan, Z.; Liu, W.; Cui, H. HIBEChain: A Hierarchical Identity-Based Blockchain System for Large-Scale IoT. In *IEEE Transactions on Dependable and Secure Computing*; IEEE: New York, NY, USA, 2023; Volume 20, pp. 1286–1301. [CrossRef]
16. Mohanta, B.K.; Dehury, M.K.; Kalidindi, S.V. Identity Management in IoT using Blockchain. In Proceedings of the 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 3–5 October 2022; pp. 1–6. [CrossRef]
17. Siris, V.A.; Dimopoulos, D.; Fotiou, N.; Voulgaris, S.; Polyzos, G.C. OAuth 2.0 meets Blockchain for Authorization in Constrained IoT Environments. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT'19), Limerick, Ireland, 15–18 April 2019; pp. 364–367. [CrossRef]
18. Tong, F.; Chen, X.; Huang, C.; Zhang, Y.; Shen, X. Blockchain-Assisted Secure Intra/Inter-Domain Authorization and Authentication for Internet of Things. *IEEE Internet Things J.* **2022**, *10*, 7761–7773. [CrossRef]
19. Polychronaki, M.; Kogias, D.; Patrikakis, C. Identity Management in Internet of Things with Blockchain. In *Blockchain based Internet of Things*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 209–236. Available online: https://link.springer.com/chapter/10.1007/978-981-16-9260-4_9 (accessed on 27 September 2023).
20. W3, "Decentralized Identifiers (DIDs) v1.0", W3.org, 2021. Available online: https://www.w3.org/TR/did-core/ (accessed on 18 August 2021).
21. W3, "Verifiable Credentials Data Model 1.0", W3.org, 2021. Available online: https://www.w3.org/TR/vc-data-model/ (accessed on 18 August 2021).
22. Reyna, A.; Martín, C.; Chen, J.; Soler, E.; Díaz, M. On blockchain and its integration with IoT. Challenges and opportunities. *Futur. Gener. Comput. Syst.* **2018**, *88*, 173–190. [CrossRef]
23. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 618–623. [CrossRef]
24. Huh, S.; Cho, S.; Kim, S. Managing IoT devices using blockchain platform. In Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT), PyeongChang, Republic of Korea, 19–22 February 2017; pp. 464–467. [CrossRef]
25. Novo, O. Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet Things J.* **2018**, *5*, 1184–1195. [CrossRef]
26. Tapas, N.; Merlino, G.; Longo, F. Blockchain-Based IoT-Cloud Authorization and Delegation. In Proceedings of the 2018 IEEE International Conference on Smart Computing (SMARTCOMP), Taormina, Italy, 18–20 June 2018; pp. 411–416. [CrossRef]
27. Li, Z.; Hao, J.; Liu, J.; Wang, H.; Xian, M. An IoT-Applicable Access Control Model Under Double-Layer Blockchain. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *68*, 2102–2106. [CrossRef]
28. Mishra, R.K.; Yadav, R.K.; Nath, P. Blockchain-Based Decentralized Authorization Technique for Data Sharing in the Internet of Things. In Proceedings of the 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 22–23 October 2021; pp. 1–6. [CrossRef]

29. Chen, E.; Zhu, Y.; Zhou, Z.; Lee, S.-Y.; Wong, W.E.; Chu, W.C.-C. Policychain: A Decentralized Authorization Service with Script-Driven Policy on Blockchain for Internet of Things. *IEEE Internet Things J.* **2021**, *9*, 5391–5409. [CrossRef]
30. Lawton, G. Top 9 Blockchain Platforms to Consider in 2022. SearchCIO. 2022. Available online: https://www.techtarget.com/searchcio/feature/Top-9-blockchain-platforms-to-consider (accessed on 27 September 2023).
31. Macdonald, M.; Liu-Thorrold, L.; Julien, R. The Blockchain: A Comparison of Platforms and Their Uses Beyond Bitcoin. *Work. Pap* **2017**, 1–18. [CrossRef]
32. Lao, L.; Li, Z.; Hou, S.; Xiao, B.; Guo, S.; Yang, Y. A Survey of IoT Applications in Blockchain Systems. *ACM Comput. Surv.* **2020**, *53*, 1–32. [CrossRef]
33. Suvitha, M.; Subha, R. A Survey on Smart Contract Platforms and Features. In Proceedings of the 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 19–20 March 2021; pp. 1536–1539. [CrossRef]
34. Ara, T.; Shah, P.G.; Prabhakar, M. Internet of Things Architecture and Applications: A Survey. *Indian J. Sci. Technol.* **2016**, *9*, 106507. [CrossRef]
35. Han, S.; Kim, J.; Lee, H.; Hwang, E. Signature Analysis of SRAM-PUF for IoT Decentralized Identifier in Large-Scale Networks. In Proceedings of the 2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN), Paris, France, 4–7 July 2023; pp. 103–105. [CrossRef]
36. Tcydenova, E.; Seok, B.; Cho, M.; Lee, C. Decentralized Access Control for Internet of Things Using Decentralized Identifiers and Multi-signature Smart Contracts. In Proceedings of the 2022 International Conference on Platform Technology and Service (PlatCon), Jeju, Republic of Korea, 22–24 August 2022; pp. 66–70. [CrossRef]
37. Yoon, D.; Moon, S.; Park, K.; Noh, S. Blockchain-based Personal Data Trading System using Decentralized Identifiers and Verifiable Credentials. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 20–22 October 2021; pp. 150–154. [CrossRef]