Received: 31 October 2018 | Revised: 3 February 2019 | Accepted: 7 February 2019
DOI: 10.1002/cpe.5238

WILEY

SPECIAL ISSUE PAPER

ControlChain: A new stage on the IoT access control authorization

Otto Julio Ahlert Pinno[1,2] | **André Ricardo Abed Grégio**[1] | **Luis C.E. De Bona**[1]

[1]Department of Computer Science, Federal University of Paraná, Curitiba, Brazil
[2]Department of Information Technology, Federal Institute of Mato Grosso, Cuiabá, Brazil

Correspondence
Otto Julio Ahlert Pinno, Department of Computer Science, Federal University of Paraná, 81531-980 Curitiba-PR, Brazil.
Email: ojapsilva@inf.ufpr.br

Summary

The IoT is changing the way we interact with the world. Very soon, almost all of our daily tasks will be made through self intelligent systems embedded in devices scattered all around us. Their mission is to turn our cities, transportation systems, buildings, homes, and bodies in smart environments. These environments will bring us more comfort, improve our performance, increase our profits, and take away time-consuming tasks. However, besides its great benefits, the IoT is also a big source of concerns, mainly because a good part of its devices will handle private and confidential information. Recently, cases of successful IoT invasions only worsen this scenario and show us that the today's adopted access control systems need to be replaced by more efficiently and secure ones. To overcome these access control problems, in this work, we present the ControlChain. The ControlChain is an access control authorization architecture that is heavily based on Blockchain technology. We also demonstrate the viability of the ControlChain through the E-ControlChain, a proof-of-concept developed to run over the Ethereum network. Our proposals follows the IoT tendency requirements and are user-transparent, user-friendly, fully decentralized, scalable, fault tolerant, and compatible with a wide range of today's access control models already used in the IoT. Finally, we also make a cost and a performance analysis of E-ControlChain, using a Raspberry Pi as an IoT device.

KEYWORDS

access control, authorization, blockchain, IoT

1 | INTRODUCTION

The Internet of Things (IoT) is a network of physical devices with embedded technology to sense and interact with their internal state or the external environment.[1] These devices are capable of collect, exchange, process, and store data in a programmatic way. Although a lot is discussed about the correctness of the predicted numbers of IoT devices for the next years,[2] even the most conservative forecasts are predicting tens of billions of IoT devices and a produced data with a potential economic value of USD 11 trillion.[3]

The IoT emerged with the objective of providing new intelligent services and commodities to facilitate our daily tasks. Its devices are pervading our cities, public buildings, roads, airways, factories, retail stores, offices, hospitals, homes, and bodies.[4] Furthermore, with their sensors, communication and information processing capabilities, they affect our interactions on all application domains, ie, personal, home, government, utilities, enterprise, and industry,[5] and create the so known smart-everything, like smart homes, smart grids, smart cities, and so on. However, together with the great features that arise with such integrated systems, there are many security and operational concerns that hinders its broadly adoption by users, governments, and industries.

The main IoT adoption concerns are about privacy, technological constraints, social and economical aspects, confidentiality and integrity, reliability and availability, and usability.[5] All of them have different level of importance depending of the application domain, eg, an industry could be more concerned about reliability and availability than a mayor of a smart city regarding a park temperature monitoring system. However, although it does not receive all the attention that it deserves,[6] the privacy concern has a big role in the prevention of IoT broadly adoption.

The access control is one of the most important tools to prevent unauthorized access and privacy invasion. Therefore, it is directly or indirectly related to a wide range of the concerns turning around the IoT adoption. In order to grant privacy in IoT, the access control needs to be capable of defining who, when, and how someone is authorized to use or access a device or its information. If it is faulty or misconfigured, it could cause a big harm to those depending on it.

Back in 2014, research studies were capable of hacking and disabling a car transmission and breaks.[7] This is one example of the vast collection of improper accesses exposed in the work of Greenberg and Zetter.[8] In 2016, security breaches on access control systems lead to more than 150 000 worldwide IoT devices being compromised and used on Internet attacks.[9] More recently, the VPNFilter malware[10] infected over 500 000 routers around the world by exploiting known access control vulnerabilities of these devices. All these successful attacks keep remembering us how the currently adopted access control systems are ineffective. It also proves that, although the access control is an old discussion, it still requires a lot of attention, specially now, that we are entering in the IoT era, where all the things are planned to be connected and, in some way or another, exposed to the world.

A complete access control solution involves three components,[11] namely, authentication, authorization, and auditing. The authentication identifies the correctly identity of the requester. The authorization (also simply known by ''access control'') verifies if the requester has the rights to do some operation on the resource. Finally, the auditing (or accountability) allows the posterior analysis of the realized activities. All these components have important roles in securing the system; however, the authorization requires special attention because it is responsible for enforcing the access rules.

Design a suitable authorization mechanism that is capable of dealing with all the complex characteristics of the IoT is a challenging task. Some of these characteristics, like the access control in big networks,[12] were already studied in other environments that held them separately. However, they were never studied all together before the advent of the IoT.[5] Common examples of such characteristics are the network size, network connections dynamism, heterogeneity of devices, and the data sensitivity.

Most of the research studies in the IoT access authorization field employ three traditional and well-known architectures, namely, XACML, OAuth, and UMA. However, all these three architectures are centralized by design and, therefore, fail to provide essential IoT access control requirements, like scalability, transparency, and resilience for the authorization process. Changes in its design are required to turn them into a suitable solutions for IoT. One of these works[13] tried to reduced the gap between these architectures, specifically the XACML, and the IoT environment. They externalized the Policy Decision Point (PDP) to a virtually unlimited resource server. However, it still does not provide transparency and resilience, which we believe to be essential for the broadly adoption of the IoT. Therefore, even with a lot of effort to bring a suitable access control to the IoT,[5] there are design barriers in the traditional architectures that prevent them to achieve a complete success.

With the objective of turning the authorization process more decentralized, user-driven and, consequently, achieving more privacy for IoT users, the works of Ouaddah et al[4,14] started to employ a disruptive technology, namely, Blockchain, in the access control. The Blockchain[15] is composed by a set of mechanisms that allows it to work as a decentralized database where, eventually, all the networks nodes end up with the same database records.

In a more specific definition, the Blockchain is a public, decentralized, and Byzantine fault-tolerant ledger, where registers are appended in a chronological order and become more immutable each time a new register is appended to it.[16] All the network nodes that participate in the Blockchain are free to verify the legitimacy of all published (appended) data, have a copy of the entire database, and publish its own data that, of course, will also be verified by the entire network.[17] Every node, or a good part of them, having a copy of the Blockchain, ensures no downtime. All network nodes having to verify the published content provides no fraud. Finally, each node being capable of publishing its own data grants no censorship and no third-party interference. Actually, these problems could happen; however, there are pretty good incentives to keep all the network nodes honest and, with the majority of them behaving in a honest way, it is extremely improbable that a small group of nodes can launch a successful attack.

Seeing all this potential, South Korea is investing more than USD 200 million in the development of Blockchain-based projects.[18] The supported projects focus on electronic document distribution, marine logistics, easy real estate, online voting, personal customs clearance, and management of livestock records. Besides this, there are plannings to support also Blockchain as a Service (BaaS) solutions. All this effort is to bring more legislative clarity and transparency.

FairAccess[4,14] proposed a Blockchain-based access control framework, where the Blockchain is used to publish smart contracts that provide valid access tokens (secrets that grant access to a resource). However, as will be discussed later, their proposal also has big issues like the support to token-based authorizations only (not being compatible with a lot of already adopted IoT access control models), necessity of contact with the owner of the resource for each new access or renew of expired token, the highly time cost involved in getting an access permission, or renewing one, and the lack of integration of the access control with a proper relationship network that has a big importance in a collaborative and integrated IoT. Our proposal promises to tackle these issues and bring improvements to the authorization process.

The usage of Blockchain also brings a few drawbacks. We consider as the worst of them the following problems: information recently appended can be more easily modified or deleted from the Blockchain when compared with the more old ones; normally, the process of appending an information to the Blockchain requires the generation of a proof-of-work, which consumes a lot of computing power; furthermore, all the network nodes have to verify all the published information before trust in it. These last two items can be a problem in nodes with low processing power or energy supply; however, as we will see, our solution also brings directions to the minimization of these effects on them. Finally, even with these disadvantages, we believe that the benefits of the Blockchain are bigger than its ill effects for a wide range of applications and environments.

In this work, we present the ControlChain, an authorization architecture that is heavily based on the Blockchain technology. It uses the Blockchain as a platform for storing and synchronizing all the data essential for authorization decision making. Furthermore, we also expanded the original ControlChain[19] to a hybrid approach. This main change was the employment of a side channel, off-Blockchain, used for the propagation of time-sensitive data. Our architecture overcome the FairAccess and traditional architectures problems with a complete decentralized and transparent authorization process that also brings legacy compatibility with a wide variety of access control models already employed on IoT (requiring minor adaptation efforts). It also includes methodologies to declare relationships between entities, ie, users and devices, and a new way to apply them in the authorizations. Finally, we design and implemented the E-ControlChain, a ControlChain implementation model for the Ethereum network, and analyzed the overhead and burden of running it on IoT devices.

The contributions of this paper are the explanation, discussion, and presentation, in a comprehensive way, of the following topics:

- A new phase-based way to explain the under-the-hood Blockchain main steps.
- A compiled tendency list of the essential and required characteristics for the IoT access control.
- The ControlChain, an authorization architecture that emerged from the IoT access control requirements.
- The E-ControlChain, a ControlChain implementation model that was designed to work over the Ethereum network.
- Finally, a performance analysis of the E-ControlChain running on an IoT device.

The remaining of this work is divided as follows. Section 2 presents some knowledge required for the rest of the work. Section 3 shows the related works. Section 4 describes the ControlChain, our proposed architecture. Section 5 presents the E-Controlchain, the ControlChain implementation model. It also shows the results of running it over an IoT device. Finally, Section 6 concludes this work.

2 | BACKGROUND

As stated before, the access control is, if not the most, one of the most crucial topics to grant the broadly adoption of the IoT. A good access control, together with a good configuration and an easy-to-use management interface, can prevent privacy breaches and unwanted accesses. We define the used nomenclature in Table 1 and show an overview of a complete access control components in Figure 1.

A complete solution for securing the access to a system is comprised of three components,[11] namely, authentication, authorization, and auditing. The authentication identifies the correctly identity of the requester. The authorization (also called simply by "access control") verifies if the requester has the rights to perform the required actions on the resource. Finally, the auditing (also known as accountability) allows posterior reviews of the performed activities in the system. In this work, we sometimes present directions on how to make authentication and auditing, but our main investigation is about the authorization process.

Developing an access control for the IoT is a challenging task mainly because it has to deal with the complexities from the IoT environment. Some of the IoT complexities are as follows: a very large number of devices, heterogeneity, unreliable communication, dealing with private information, limited processing power or battery, and so on. Therefore, in Section 2.1, we show a list of the main requirements for the IoT access control.

TABLE 1 Adopted nomenclature

Nomenclature	Meaning
Requester	The entity trying to access a resource
Action	The activity that the requester perform over a resource
Resource	A generic device or data that can be accessed or manipulated
Entity	A requester, resource or a set of them
Identity	A value that uniquely identifies an entity
Context	A set of unambiguous attribute-value pair in the environment
Context identifier	An ordered pair (entity, variable), where the variable is entity-dependent
Miner/Validator/Sealer	Who append blocks to the Blockchain

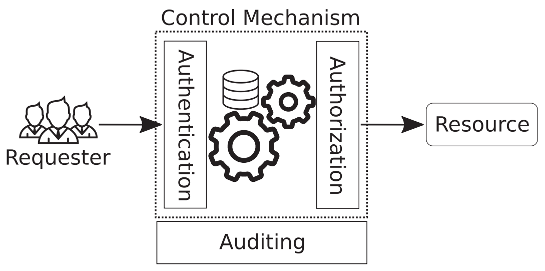


FIGURE 1 Access control components overview

2.1 | IoT access control requirements

A secure IoT access control system must be capable of (1) precisely identifying the entities with robust anti-fraud mechanisms to prevent non-authorized malicious devices or users from accessing resources, (2) enforce the access policies defined by resource owners, and should (3) have the following characteristics[5,13,14,20]: scalability, lightweight, transparent, user-friendly, fault-tolerant, privacy-friendly, delegation-capable, context-aware, and fine-grained. Furthermore, we also argue that it needs to be relationships-aware and legacy-compatible. Next, we describe why these characteristics are important to the IoT.

Scalability. Even the most conservative forecasts are predicting the existence of tens of billions of IoT devices for the next years.[2] A centralized access control architecture will easily become a bottleneck and give limits to this fast growth. In this type of scenario, the most recommended is the adoption of decentralized or distributed architectures.

Lightweight. The IoT is known to be composed by a huge diversity of devices, ranging from powerful processing ones, like smartphones, to powerless ones, like RFID sensors. All these devices will be protecting some resources and, therefore, checking the existence of authorization for each access attempting. Therefore, it is important that the architecture provide low cost operations, mainly, for checking authorization as it will be accomplished by IoT devices in most of the cases.

Transparent. A considerable part of IoT will be composed of personal and intimate devices. They will collect and manage user's private information. The more intimate is the information, more concerned the owners are about who, when, and how the information are being handled. The authorization process is required to be transparent to gain confidence from users.

User-friendly. Even in the IoT era, there are still users with almost no knowledge in computer mechanisms and systems. This means that a confused or complicated authorization system has a good chance to keep away a lot of potential IoT users or turn them into victims of unauthorized accesses. Therefore, the access control has to be user-friendly and has to have a low learning curve for new users.

Fault tolerant. The IoT will be surrounded by unreliable devices and network connections. Some examples of these instabilities are IoT devices could run out of power supply, wireless channels could suffer from interference, mobile devices can move away from each other and loose connections, etc. Thus, these and other instabilities generated by the IoT characteristics should not affect authorization mechanism.

Privacy-friendly. If not all, at least the personal and intimate information has to be strong protected. Therefore, an access control has to be based on reliable and robust mechanisms, like cryptography, that enforces access policies and hide the information from undesired accesses. This is also an important requirement to gain confidence from users.

Delegation-capable. Delegation is the forwarding of a given permission from one entity to another. However, while providing it brings a new set of functionalities, it also could be dangerous since one single entity with the permission could forward it to any number of entities. Therefore, we argue that somehow the delegation needs to be regulated or controlled by the object owner.

Context-aware. In the IoT, all things are meant to be interconnected and to exchange information. The exchanged information can help the IoT to make better decisions. For example, if there is an emergency, security devices could unlock all doors and let anyone to enter the site. Therefore, it is desirable that the access control can deal with context-aware access rules.

Fine-grained. A lot of works already saw the potential of having an access control with fine-grained control.[13,21-23] As the IoT has a wide variety of information, the fine adjustments in the access control is very welcome in complex environments that requires it.

Relationship-aware. Collaborative, integrated, and interconnected system that takes in consideration relationships is becoming more and more popular. An example of this popularity is the recently wave of games in the social networks that allows user friends to interact with him or her in its game play.[24] Probably a lot of these games would not have such acceptance if its interaction was only with completely strangers. Therefore, in a similar manner, we also see a great potential in exploring relationships in the IoT access control.

Legacy-compatible. As we will see, there is a wide range of access control models been proposed and used in the IoT. Each one has its peculiarities and we do not believe that one single model is capable of completely satisfying all the different scenarios and environments requirements. Furthermore, in order to not require the disabling of all the old models, it is desirable that a new architecture be legacy-compatible and, also, support as many models as possible.

2.2 | Blockchain

Until recently, almost all financial transactions were intermediated and centralized by third party organizations. This dependency brings several drawbacks, like the requirement of trust in the central entity and the lack of transparency in the operations. In a pursue for a secure and decentralized transaction environment, the Blockchain made its way to our world as a public, decentralized, Byzantine fault-tolerant, immutable, chronological-order registers database solution for cryptocurrencies, more specifically for the Bitcoin.[15]

Since its creation, the Blockchain popularity has been getting more and more to unthinkable levels. Although a good part of this popularity is because of its successful adoption in the cryptocurrencies, it also has proved its capability to revolutionize many other areas like access control, IoT, smart contracts, smart property, digital content distribution, botnet, P2P broadcast protocols, and so on.[4,25]

The Blockchain is a multi-field construction composed by wide variety of technologies, like cryptography, mathematics, economic model, peer-to-peer networks, and algorithms.[26] Together, they create a solid, secure, and decentralized database that stores data records in an ordered and crescent list, commonly named as ledger.[25] All the records in the ledger are public to any participant of the Blockchain. Thus, anyone can have a copy of the entire Blockchain with all the information ever published on it.

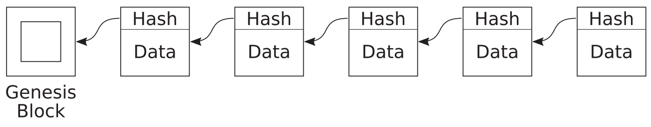


FIGURE 2 Blockchain main structure

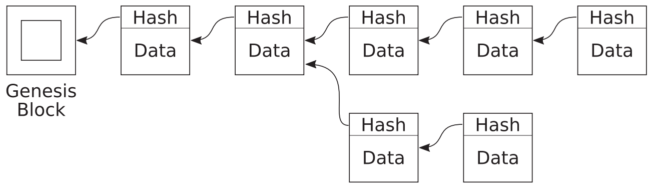


FIGURE 3 Blockchain main structure with branches

In a more technical view, the Blockchain is an ordered sequence of blocks. These blocks are composed by a predefined amount of information and by a fundamental field that holds a link to the immediately prior block in the sequence, thus, creating a chain of blocks (see Figure 2). The information contained in these blocks are flexible and only depends on the system requirements where the Blockchain is being used.

In order to grant the security of the applications running over the Blockchain, some systematic safety instructions have to be followed by all participants in the network. If one of them chooses not to follow these instructions, it will only be ignored by the rest of the network, causing no harm to it. Therefore, all participants have the incentive to continuously follow the safety instructions and collaborate with the network. For the explanation of these instructions, we divided the Blockchain usage in four phases, namely, data creation, block construction, block appending, and consensus reach.

The data creation phase is the process of creating a message for publication in the Blockchain. It needs to be built according to the application protocol in order to be accepted by the network. Therefore, the rules for its creation are exclusively made by agreements between the participants of the application running over the Blockchain network. Normally, it is required signatures and data that could be validated. After its creation, it is broadcasted to the network.

The messages generated by the data creation phase are used in the block constructions phase. It encompasses receiving the messages generated in the data creation phase, checking its correctness, grouping the correct messages in blocks and, finally, mining it. This process is done by special network entities, namely, miners. In fact, any participant of the network can be a miner. Moreover, depending on the adopted "mining" consensus methodology, the miners and the process of mining can receive different names, eg, miners can be called validators or sealers and the process as validation or sealing. Of course, the way that the "mining" occurs is different in each one. The original one consists on finding a nonce that causes the hash of the block to be less or equal a predefined value. After it is mined, the block is broadcasted to the network.

When a new mined block is received, the participant has to verify if all the messages in the block are in accordance to the application protocol. Besides this, it also has to check if the block follows the rules defined by the application. If the block does not pass on any of these validations, the participants reject the block. Otherwise, they append it to their local Blockchain.

Eventually, the Blockchain also can fork and give origin to two or more branches, forming a tree-like structure. A new branch arises when the new received mined block, instead of pointing to the last known block, points to a prior block (see Figure 3). This can happen by uncountable reasons, like delay of propagation, bipartition of the network because of link problems or malicious participants, or even blocks being mined at the same time. If the Blockchain application does not support branches, it also is required to implement consensus algorithms to deal with synchronization problems and resolve disputes. A commonly used consensus is to adopt the longest branch and forget the others.

Particularly, for the access authorization process, we consider the time for a new information be part of the Blockchain as its main drawback. Of course, this is mainly dependent on the type of proof, its level of difficult, the complexity of the adopted consensus establishment approach, and the amount of produced and mined information per unit of time. However, even so, we still consider it as a mechanisms that brings more benefits than drawbacks to the IoT authorization process, especially when new Blockchain-like mechanisms are emerging, eg, Algorand[27] and DAG-based ones.[28,29] These mechanisms can drastically reduce costs and mining time.

2.3 | Ethereum

The Ethereum[30] is an open-source and public Blockchain network that provides Turing complete distributed computing through smart contracts. Therefore, Ethereum smart contracts support branching and looping statements, and also state variables. Thus, they can be used to solve any problem that can be solved by computers.

Interactions in the Ethereum network are done using identifiers called addresses. Each user and contract has its own address. User addresses are derived from their public key. Contract addresses are computed using its creator address and a nonce, the transactions number of its creator. Both cases uses, as addresses, the last 20 bytes from the result of the keccak-256 (SHA3-256) over the public key (for users addresses) or the creator address and the nonce (for contracts). Although address collision is possible, it is extremely improbable as there are 2^{160} different possibilities of addresses that can be generated with this methodology. Nowadays, there are already more than 40 million distinct addresses on the network and this number is trending up.[31]

Compiled and published contracts are called "Decentralized Applications" (or DApps). Contract publications or interactions are charged in a unit called Gas. This unit is a measurement of the effort of doing the user request action. Each low-level operation has its own fixed Gas cost and the amount used in an action is the sum of the expended gas in each operation. The price of Gas (called GasPrice) fluctuates over time and is given in Ethers (the Ethereum cryptocurrency). There are already more than 1000 DApps launched since 2017 and, just taking in to account two of these contracts, there are more than 1000 daily active users,[32] ie, users that interact daily with, at least one of these two contracts.

3 | RELATED WORK

Although the division of access control approaches in different layers is not always straightforward, there were already efforts trying to standardize this classification based on the OM-AM reference model.[5] They classify the state-of-art in four layers, namely, Objective, Model, Architecture, and Mechanisms. In the objective layer, the access control policy (high level rules) is investigated. The Model layer defines unambiguous means for apply the access control policy. The Architecture layer describes the entities of the access control, their workflow and interactions. The mechanism layer defines the software and hardware used in the access control policy enforcement. Next, we choose to present only works that are somehow related to the architecture layer because our work is more related to this layer.

FairAccess[4,14,33] is an access control framework based on smart contracts, Blockchain, and token-based access. The smart contracts are used to trade fulfillment of access control policies for access tokens using the Blockchain. Mainly, it consists of four steps. (1) The user makes an off-line solicitation of access to the resource owner; (2) the resource owner publishes an smart contract that defines an access token and the requirements to this access token be accessible by the user; (3) the user fulfill the requirements and publishes the proof; (4) finally, it can access the device with the access token. Different from our proposal, the disadvantages of their solution are fourfold. First, necessity of contact the owner of the resource for each new access or renew of expired token. Second, for a new token be usable, it is required at least two blocks to be mined to the Blockchain, ie, it is costly and could take a lot of time to the access be granted. Third, they only support token based authorizations. Fourth, they do not give any solution for allowing the usage of relationships in the process of granting access to a subject. The only relationship information that could be inferred is as follows: with two identities (public keys), it is possible to know if one is parent of the other in the tree of the generated keys. In our proposal, identities can be linked with any other identity and each link can have attributes and characteristics attached to it.

In the work of Zhang et al,[34] another framework for IoT access control using Blockchain was presented. It is composed of three main contracts types, ie, Access Control Contract (ACC), Judge Contract (JC), and Register Contract (RC). The ACC define the access policies through `resource`, `action`, `permission`, and `time of last access` fields. However, each ACC only supports a single requester-resource pair. Depending on the use case and the network, this design choice can elevate the cost to near prohibited ones as each new pair requester-object would require a new contract to be published. On the contrary, we propose that all users use a single contract and dynamically define access rules through it. The JC is the contract that judges misbehavior of the requesters and define their penalties. The RC is a lookup table that keeps the required information to find and execute all the methods of all contracts.

Other works[35-37] highlighted the benefits from the union between Blockchain and IoT. In addition, the works of Kumar and Mallick[35] and Panarello et al[37] discussed the challenges of IoT networks and its integration with the Blockchain. Panarello et al[37] also constructed a systematic survey, where they categorized a wide range of works in application domains, usage domains and development level. Nuss et al[36] discussed the benefits of Blockchain adoption in an Identity and Access Management (IAM) in an enterprise context. It also proposed an architecture similar to our previous work.[19] However, there are also some differences. First, they only bring support to attribute-based access control (ABAC) model. Second, they do not provide alternative solutions to applications that require exchange of real-time information, requiring all the data to be mined by the Blockchain in order to be available in new decisions. Finally, they did not evaluated their architecture on a real IoT device.

MedRec,[38] Bright,[39] and the work of Zyskind et al[40] also employed Blockchain in the access control. MedRec controls the access permissions to medical record data of patients. It employs smart contract between patients, providers, and third parties to grant permissions of access. Bright controls the actions performed in a video rights management system. The work of Zyskind et al[40] uses the Blockchain as a mechanism of data sharing. The data is stored in a off-Blockchain DHT network and only the pointer to this data is stored in the Blockchain. Different from our architecture, these works only allow creation of ACL-like rules and do not support other information in the authorization process, like the environment context.

IBM Watson IoT[41] is a platform that, between a wide range of services, manages and controls the access to IoT devices. It also allows device data to get published into a private Blockchain to reduce the dependence of a central management entity in the data access. However, all the configuration of the access control is centralized.

The solution proposed in the work of Dukkipati et al[42] uses the Blockchain to store, among other things, pairs of resource identification and URL links. The links are addresses of access policies stored on the proper resource device. A smart contract is used to retrieve the policy and check if the user has permission to access the resource. It has the advantage of updating the access policies locally, ie, does not require the mining of each update. However, as the policies are stored off-Blockchain, they are susceptible to modification. This could lead to security problems.

Zhu et al[43] used the Blockchain together with an ABAC model. However, in the workflow of its platform, they describe the use of the Blockchain as log database for all actions executed by their ABAC modules. In addition, no evaluation in the IoT environment was made.

FedCAC[44] proposed a hierarchical architecture for access control. It is based on two main control entities types, ie, the Policy Decision making Center (PDC) and the Coordinator. The PDC is located in the Cloud and the Coordinator is located near of resources. The PDC is a common point in the exchange of messages between Coordinators, turning it into a single point of failure. In the work of Xu et al,[45] they resolve the centralization problem with a Blockchain-based architecture called BlendCAC; however, in both works, they only support capability authorization using tokens.

Pohrmen et al[46] discussed how some of the IoT challenges could be minimized with the use of Blockchain and, also, defined a Blockchain-based IoT architecture divided in three layers, namely, infrastructure, control, and application. The infrastructure layer consists of networking technologies and elements, and a local Blockchain with resource constrained devices. The control layer is composed by a global Blockchain that have more powerful devices as miners. Finally, the services and user applications reside in the application layer.

In the work of Klaokliang et al,[47] the authors proposed an IoT authorization architecture over the Hyperledger Fabric,[48] a Blockchain implementation held by the Linux Foundation. The main contribution of their work is the enhancement of the consensus component with a genetic algorithm-based solution called GA Kafka. With this improvement, they achieve a better transaction transfer success rate.

Although Wu et al's[49] solution is more about providing authentication instead of authorization, they provided a scheme that stores relationships between requester and sensors. When a requester tries to access a data from a server, it gets a challenge as a response that needs to be accomplished to authenticate and receive the requested data. Each challenge defines a sequence of actions for the requester. Sensors near the client, namely, Related Devices (RD), have to detect and publish the client performed actions in the Blockchain. However, their solution have a big flaw, ie, the server selects the challenge RDs using the requester's published near RDs. This could lead to the selection of fake or malicious RDs. A simple solution would be letting the source of the data define the RDs or the choosing criteria for the RDs used in the authentication to access its data.

Bubbles of Trust[50] is another solution for authentication in IoT using the Blockchain. The main idea behind Bubbles of Trust is the creation of IoT devices groups where devices only trust in devices that participate from the same groups. These groups are created by a master device that distributes tickets for other nodes, called followers, to join the group. The group creation and devices joining is registered in the Blockchain.

BSeIn[51] allows requester to make request to IoT industrial devices without revealing its identity. In order to do that, they make use of a one-time key generation schema and Blockchain. In their approach, all requests need to go through the permissioned Blockchain. This design decision limits the number of requirements to IoT devices to the maximum number of requirements the Blockchain miners, validators or sealers can handle.

In the work of Molina-Jimenez et al,[52] the authors enumerated some of the challenges that can be present in proposals that adopt Blockchain as part of their solutions. In a special way, they discuss about the high cost of having all the operations performed over the Blockchain, turning them impractical. In addition, they propose approaches to minimize this cost, like externalizing, from Blockchain, some non-essential Blockchain operations. One of these approaches is called as "hybrid implementation." It separates contractual operations into d-op (decentralized operations for Blockchain interactions) and c-op (centralized operations for Trusted third party - TTP - interactions). As expected, they defend that all the operations that are not crucial to be in the Blockchain should be moved away from it and directed to the TTP. In their example, the TTP is a gateway assistant that gives insights about allowing or denying attempts of data access. In our prototype, any Ethereum node can be a "TTP" and, eg, verify the access permissions executing the smart contract locally (without any extra cost and with a similar delay of consulting a lightweight database). Of course, one should be very cautious when relying on TTPs. Furthermore, our architecture also can be classified as a "hybrid implementation" solution as it also has an off-Blockchain side channel to exchange, eg, real-time messages.

The data exchanged over a side channel can be considered more volatile than the published on Blockchains; however, some scenarios require this feature. For example, to avoid the possibility of multiple spending of a one-time voucher that can be caused by the delay of Blockchain on registering its usage. The usage of a side channel also corroborates with the so-called "Layer 2" era of the Blockchain.[53] In this era, the computation is moved off-chain to enable privacy, reduce costs, or save computing resources.

Although they are not so efficiently for the IoT as the already presented approaches, the XACML,[54,55] OAuth,[56] and UMA[57] architectures were already employed in the IoT access control. None of them bring solutions to the problem caused by the architecture components centralization. This dependency of a central entity makes them susceptible to the single point of failure problem and could limit the growing of the IoT. Therefore, their utilization leads to a less fault resilient and less scalable IoT. Furthermore, mainly because of the centralization, solutions based on them can easily lack from transparency and privacy.

4 | CONTROLCHAIN

As we saw, the previous approaches were only capable to cover a little part of the IoT requirements. Some of them do not scale, others do not take context information into consideration, do not have support to relationships, or ignore the already used IoT access control models. Trying to meet more of the IoT requirements, we came up with the ControlChain, an access control architecture heavily based on Blockchain.

The Blockchain has mechanisms that are capable of providing a good part of the IoT requirements (see Sections 2.1 and 2.2). Since its origin, it was used as a decentralized database that eliminates the necessity of a third party control. It also can employ sub-mechanisms to provide resilience to data corruption or attacks trying to remove or replace a published data. However, it is important to highlight that, with

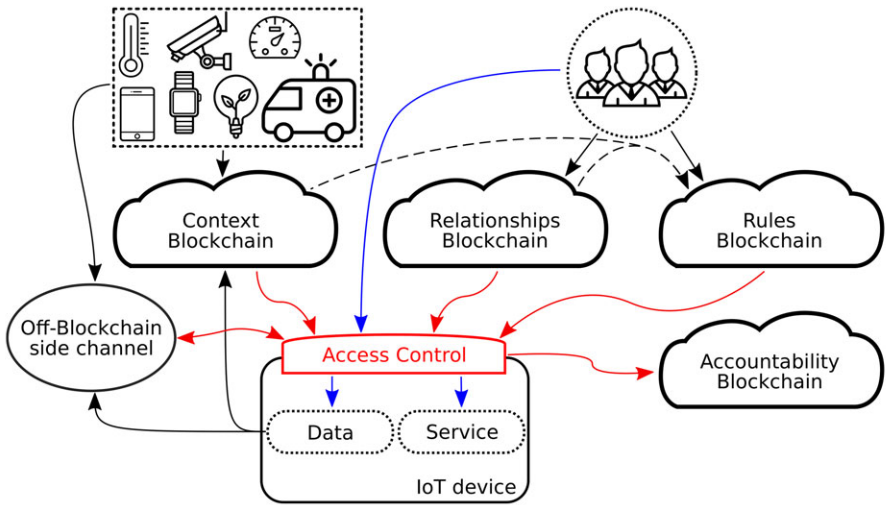


FIGURE 4 ControlChain overview

the appropriate management of the published data, it is possible to "update" the Blockchain data via the publication of new registers that takes priority over old ones and turn them in historical data only, ie, with an instruction that mark the old data as canceled and creates a new valid one. Furthermore, the way that its decentralized database works also offer the possibility of off-line authorization.

Another important requirement is the compatibility with the largest number of access control models, especially the ones adopted for the IoT. This is also possible to be achieved with the Blockchain as it does not require a strict data information or structure. The data type it can store only depends on its implementation details. Therefore, virtually, any type of authorization instruction can be stored in it.

Figure 4 presents an overview of the ControlChain architecture. Although the data exchanged through the Blockchain can be considered as less volatile, we extended our prior version of ControlChain[19] with an off-Blockchain side channel that can be used to exchange real-time data, reduce costs, and provide more privacy (if running over a trustworthy platform). Note that these are crucial features for some IoT scenarios (see Section 3). Thus, the ControlChain architecture is composed by IoT users and owners, IoT devices, real-time channel, and four Blockchains, ie, context, relationships, rules, and accountability.

Although nothing prevents all the information to be in just one unique Blockchain, for organization and to facilitate the explanation, we divided the required data of the ControlChain in four different Blockchains. The next sections bring more information about each one of these Blockchains, like their roles and the interactions behind them.

4.1 | Relationships Blockchain

The Relationships Blockchain stores the entities public credentials and all the relationships between them. An entity is a user, a device, or a group of them. In practice, there is no differentiation between them in ControlChain specification. Moreover, each entity can have a designated owner that has complete control over it. A relationship is a unilateral reference to another entity with an optional set of attributes, ie, the entity specifying the reference can give attributes to the referenced entity.

There are two possible types of relationship references, ie, Blockchain-dependent and external. The Blockchain-dependent reference is a link created with identifications tied to the Blockchain registers (eg, a pair composed by the block order number and the register order number inside the block). The external reference is a link to off-Blockchain identification (eg, a public key). The last is a dynamic reference that always is interpreted as a pointer to the most recent update of an entity in the Blockchain, if it exists there. Note that the external reference also allows referencing Blockchain outsider entities. The choice of best type of reference is use case dependent. In Table 2, we give some directions to the best choice based on the use case requirements. The "+" signal shows the most indicated type for the requirement.

TABLE 2 Most indicated reference type based on the requirements of the use case

Requirement	Identifications Types	
	Blockchain-dependent	External
If the most recently information inside a referenced entity need to be evaluated		+
If the information inside a referenced entity need to be evaluated as it was in the time of the authorization	+	
If references to entities outside the blockchain are allowed		+
If it is only allowed references to entities defined on the Blockchain	+	

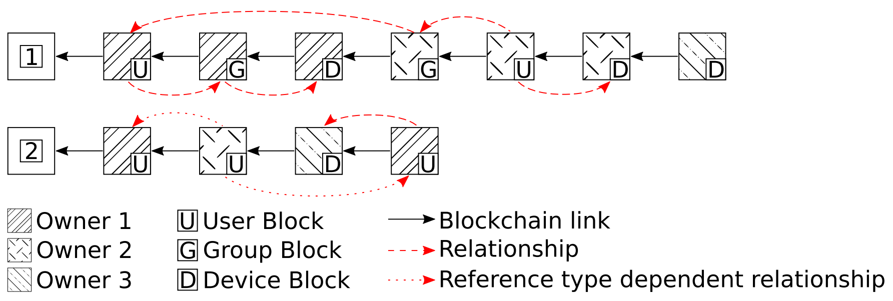


FIGURE 5 Relationship overview

Figure 5 shows two examples (numbered as 1 and 2) of relationships Blockchain. In this figure, each square is a block, the leftmost square is the genesis block, contiguous line arrows are the default links between the blocks, dashed line arrows are relationships, dotted line arrows are relationships that are dependent of the relationship references type chosen, and the hatch represents the block owner. It is important to remember that one block could have more than just one entity and not necessarily need to be labeled with their type or function like the ones in the figure. We choose this methodology only to simplify the explanation.

In the first example, the left part shows the relationship between an user, a group and a device (all owned by the Owner 1), ie, the user references the group and the group references the device. In the middle of the example, there is an Owner 2 user with a group (eg, with the attribute "friends") that links to the Owner 1 user. Finally, there could exist entities without references or being referenced like the last entity of the Blockchain. As can be seen, the type of relationship adopted in this example is the external because there is referencing of after-appended entities.

The second example of Figure 5 presents an example of a behavior that depend on the chosen reference type. In this example, the Owner 1 user was created on the second block with no relationship and was referenced by the Owner 2 user in the third block. After, the Owner 1 user was updated to reference the Owner 3 device. With the Blockchain-dependent reference type, the Owner 2 user stays referencing the first version of the Owner 1 user, ie, the one without references. With the external reference type, the Owner 2 user relationship is automatically pointing to the updated Owner 1 user, ie, the one with the reference to the Owner 3 device.

In the last example, it is also possible to see how the update of the relationships information is handled by ControlChain. When an entity publishes a new version of its relationships, the old one becomes historical data and is only used if the reference type chosen was Blockchain-dependent. In the contrary, only the new reference is used in decisions. For example, in the beginning, the Owner 1's user block has no reference; however, in the end, it was referencing a device. Therefore, in practice, there is only one Owner 1's user block (the one that references the device) and a historical data from it referencing no one.

4.2 | Context and accountability Blockchain

The context Blockchain stores contextual information obtained from sensors, processed data, and manual inputs. These contexts can be used in the authorization decision. For example, suppose there is an access rule with the following statement: "8k resolution videos can only be accessed when the router reports that the network traffic is low." In this situation, the access control will find the report of the router in the context Blockchain and check its state before allowing the access. In order to allow its reference from a rule, each context information needs to be identifiable by an unique identifier, namely, context identifier (it could be, eg, an ordered pair "(entity, variable name)"). Moreover, each time a context needs to be updated, the same context identifier has to be used to ensure that previous defined rules can find the new context value.

Accountability information can also be considered as context; however, in our specification, it has more specific information than that registered in the context Blockchain. The accountability Blockchain has information about access permissions and denies and actions performed on objects. The information that has to be registered in this Blockchain is described in the rules Blockchain, as will be explained in the Section 4.3. Between the main uses of these information are the accountability and auditing of accesses, and the sanity checking of the access control system.

4.3 | Rules Blockchain

The rules Blockchain keeps the authorization rules defined by owners to their objects or by objects to themselves. The big challenge faced by this Blockchain is making it generic enough to be compatible with the big variety of access control models and mechanisms used in the IoT, like RBAC,[58-61] ABAC,[62] UCON,[63] CapBAC,[64-67] ACL,[38-40] and others.[68-70] Each one is capable of fulfilling different IoT scenarios requirements.

Without loss of generalization, we identified three types of access control mechanisms (based on ACL, Capability, and Attribute) that, with some minor additions, could lead to the compatibility with a lot of models. These minor additions are the addition of context conditions, obligations, and accountability information. The context conditions are Boolean expressions that are built using the context identifiers of the context Blockchain. The obligations determine routines (eg, accept an agreement) that the subject must accomplish to get the access authorization. Finally, the third addition describes the access information that should be recorded on the accountability Blockchain by the access control. We named the mechanisms-based blocks as ACL rule block (allows a list of subjects for each object), capability rule block (allows a list of objects for each subject), and attribute rule block (allows a list of subjects' and objects' attributes). Each authorization rule needs to be uniquely identified (by an identifier or a set of them) to allow their updating or revoking in a similar manner as occurs in the relationships Blockchain (see Section 4.1). Section 4.3.1 gives more detail on how these type of blocks can be used to bring compatibility to a large variety of models.

4.3.1 | Decoder: model adaptation for our architecture

The transformation of access control models to more generic mechanisms is illustrated in Figure 6. We propose the utilization of a decoder. This decoder receives the access control model and its rules and translates them to mechanisms supported in our architecture. In some cases, the IoT owners also need to provide additional information. For example, suppose a rule says that only a requester with the role "manager" can have access to the management system. This role can be seen as an attribute of the requester and, at least, one entity should be pointed as the official attribute authority for it.

Table 3 presents an ease way to automate the adaptation of IoT authorization models to the three suggested mechanisms-based blocks for the Rule Blockchain, ie, ACL, Capability, and Attribute. To this end, it defines a mapping table between each element of the RBAC,[71] OrBAC,[72] ABAC,[73] UCON,[74] and CapBAC[75] models to those in the ControlChain. Note that the UCON model requires that the object constantly monitor the Blockchain for changes that violates the rules and obligations of the current accesses and interrupt any access with violated rules and obligations.

4.4 | Comparison of ControlChain

In this section, we compare ControlChain with other architectures. The comparison of ControlChain with the XACML, OAuth, UMA, and FairAccess is presented in Table 4. After, we justify each evaluation.

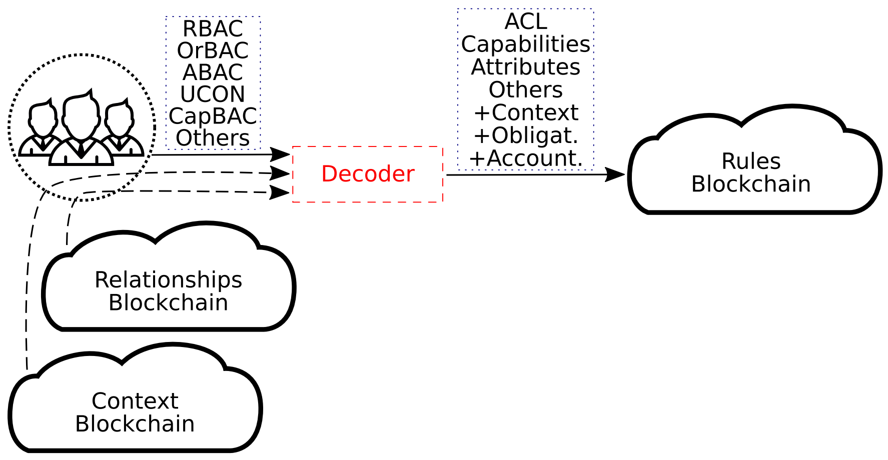


FIGURE 6 Transformation of access control models to the mechanisms

TABLE 3 Mapping of the models to the suggested rule blocks

ControlChain	RBAC	OrBAC	ABAC	UCON	CapBAC	ACL
Entity	Requester	Requester	Requester	Requester	Requester	Requester
Entity attribute	Role	Role	Requester attribute	Requester attribute	-	-
Entity	Resource	Resource	Resource	Resource	Resource	Resource
Attribute Rule Block[a]	Rule	Rule	Rule	Authorization	-	-
-[b]	-	Activity	-	-	-	-
Entity attribute	-	View	Resource attribute	Resource attribute	-	-
Entity	-	Organization	-	-	-	-
Context identifiers	-	Context	Context	Context	-	-
Cap. Rule Block	-	-	-	-	Rule	-
ACL Rule Block	-	-	-	-	-	Rule

- The model/architecture does not directly support this element.
[a]When applicable, it requires the attribute authority, subject and object attribute names and expected values, allowed actions, objects with the allowed actions, contexts, and obligations.
[b]The interpretation of activities needs to be carried by the requester and resource.

TABLE 4 Architectures comparison

	XACML	OAuth	UMA	FairAccess	ControlChain
Scalability	-	-	-	+-	+-
Fault tolerant	-	-	-	+-	+
No third-parties	-	-	-	+	+
New/Update authorization	+	+	+	-	-[a, b]
Get authorization	+	+	+	-[a]	+
Integr. relationship	-	-	-	-	+
Compatibility	+	-	-	-	+
Low object overload	+	+	+	+	+

[a]Dependent of the type of proof and dissemination speed of blocks.
[b]If using the side channel for publication of rules, it would be a plus sign.

Scalability. FairAccess and ControlChain use Blockchain as its underline technology. Although it grants scalability in the number of nodes participating in the network, it does not scale in the same manner with the number of published data. Therefore, FairAccess and ControlChain received a neutral evaluation, while the centralized architectures (XACML, OAUTH, and UMA) receive negative evaluation.

Fault tolerant. This criterion evaluates the impact caused by failures on devices or communication links. Naturally, the decentralized ones are more fault tolerant than the centralized ones. However, the FairAccess requires that the resource owner publishes a token every time an access is requested or need to be renewed. If the owner is not available to provide it, the access cannot be established. This gives ControlChain a slight better evaluation.

No third parties. The dependence on third parties could prevent the detection of censorship, frauds, and interference. All the centralized architectures depend on third parties and, thus, only FairAccess and ControlChain receives positive evaluations.

New Authorization. This criterion evaluates the latency to make or change an authorization. The centralized architectures XACML, OAuth, and UMA have low latency in these activities because the update of their database is straightforward. Thus, they received a positive evaluation. For FairAccess and ControlChain, the complete analysis of this criterion is dependent of the blocks dissemination speed and, mainly, the type of proof used in the blocks mining. As the most common and known type is the proof-of-work and this type of proof imposes a considerable latency, we give a negative evaluation. It is important to note that the FairAccess does not specify a way to modify a contract already published on the Blockchain. ControlChain specification defines a way to update the information (see Sections 4.1, 4.2, and 4.3). Moreover, if the ControlChain side channel is used to publish rules at the same time it goes to be published on Blockchain, the time for sharing a rule or information could be very similar to the centralized approaches, and thus it would receive a positive evaluation.

Get authorization. This criterion evaluates the latency to get an authorization. XACML, OAuth, UMA, and ControlChain have almost real-time authorizations. The FairAccess has a bigger latency because it requires that the requester publish a fulfillment proof of the contract requirements. This requires, at least, one additional block to be mined to the access be granted. Thus, it is the only one that receives negative evaluation.

Integrated relationship. The ControlChain is the only architecture designed to directly allow relationships on rules. Therefore, it was the only one positively evaluated.

Compatibility. This criterion evaluates the compatibility of the architectures with the plenty of models currently employed in the IoT. XACML and ControlChain are compatible with a considerable quantity of models, and then they received a positive evaluation. Indirectly, OAuth, UMA, and FairAccess can operate with almost any model; however, directly, they only operate with access tokens and, thus, they were negatively evaluated.

Low object overload. This criterion evaluates how much the resource is overloaded by the authorization process. In the centralized architectures, all the authorization process is externalized to a powerful entity by design. However, nothing prevents the FairAccess and the ControlChain to externalize their authorization process too. In fact, the ControlChain can facilitate the automation of this process when it is necessary. A device could search, in the Relationships Blockchain, for other confident devices, eg, from the same owner, and ask their support in the authorization process. Therefore, all the architectures were positively evaluated.

5 | E-CONTROLCHAIN

The E-ControlChain (Ethereum-ControlChain) was created as a ControlChain proof-of-concept to check ControlChain practical viability for the IoT. The E-ControlChain is a contract designed to run over Ethereum platform. It was developed using Solidity,[76] a widely known and used smart contract language for the Ethereum Blockchain network. In order to make ControlChain compatible with it, we performed some adaptations, eg, all the interactions with the ControlChain had to be through function calls and we have not implemented all of its definitions (more details in Section 5.2). In addition, we focused on the Blockchain interactions only, so we did not implement the side channel as it can be done using well known protocols, eg, XMPP or MQTT.

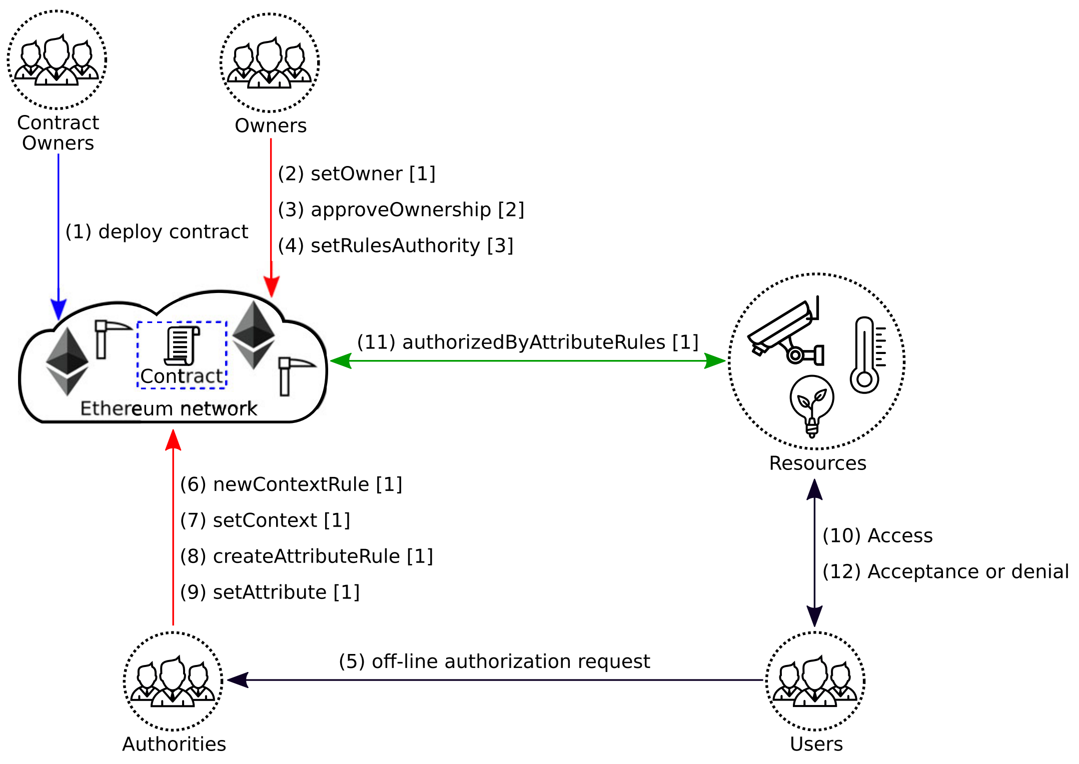


FIGURE 7 E-ControlChain interactions

The E-ControlChain is composed by four contracts. The first one, called `BasicAndCommonControl`, defines variables and functions common to all other contracts. The other three are specialized in each type of authorization defined by the ControlChain architecture, ie, authorization based on attributes (`AttributeControl`), capabilities (`CapabilityControl`), and ACL (`AclControl`). For brevity, in this work, we will present only the essential contract parts for the attribute-based authorization, which involves both `BasicAndCommonControl` and `AttributeControl`. Its pseudocode is available on the Appendix.

Figure 7 presents all the crucial interactions that are available for the E-ControlChain attribute-based access control (red and green arrows), together with the contract deployment (blue arrow) and the expected off-line and direct interactions (black arrows). In the figure, each interaction has a number; however, except by the prerequisites (defined in the box brackets), there is no strictly execution order. Thus, one can invoke `(7) setContext` before it invokes `(6) newContextRule` because they are independent functions. However, it is impossible to invoke `(4) setRulesAuthority` before `(3) approveOwnership` because `(4)` depends on `(3)`.

Before being used, E-ControlChain needs to be deployed. The deployment is made sending its compiled code to the Ethereum `(1)`. It has to be done only one time. All the E-ControlChain function calls can be directed to the same deployed contract. There are eight fundamental function calls to provide the ControlChain attribute-based access control: `(2) setOwner`, `(3) approveOwnership`, `(4) setRulesAuthority`, `(6) newContextRule`, `(7) setContext`, `(8) createAttributeRule`, `(9) setAttribute`, and `(11) authorizedByAttributeRules`. Keep in mind that, for brevity, some functions dedicated to "update" or "remove" published contents are not shown. For example, there is also a function `deleteAttributeRule` that we will not discuss. Furthermore, some of the presented functions, like `(7) setContext` if using a previous used context identifier, can be used to update the access control information. Thus, the update of some information does not require a special function.

The `(2) setOwner` and `(3) approveOwnership` functions define the ownership of an entity and approve it. If it already has an owner, only the already defined owner or the entity itself can change its ownership. Moreover, in order to avoid malicious entities from pretending to belong to an entity they do not belong to, only the defined new owner can approve its ownership. After the ownership is approved, the owner can delegate the access control over the resource to another entity using `(4) setRulesAuthority`.

The rules authority is responsible for creating the access control rules for its delegated resources. It creates the attribute-based rules using the `(8) createAttributeRule` function. Each rule defines the required attributes for the resource and requester, and its respective attribute authorities, ie, the entity that has to provide the attribute. Besides this, it also defines the allowed actions and a list of already created context rules that needs to be true in order to the access be granted.

A new context rule can be created with the `(6) newContextRule` function. Each context rule is an ordered quadruple `(source, identifier, comparator, value)`. It defines that the `source` is the provider of the context identified by `identifier` and that, in order to allow access, the comparison, using the `comparator`, of the current context value with the `value` has to be true. Currently, the `comparator` can be any common relational operator <, <=, ==, ! =, >= or >; however, it can be adapted to support other comparators. Note that a tuple `(source, identifier)` is a context identifier. The context rule always uses the more recently published context and only the `source` itself or its owner can publish/update its contexts. A context is published with the `(7) setContext` function. This function requires the context identifier and a value for it.

An authority also can give attributes to an entity using the `(9) setAttribute` function. The given attributes are isolated for each authority, ie, the ones given by an authority does not get mixed out with the attributes given by another authority. Note that all the rules that use information from authorities have to provide the identity of the authority from which the information will be pulled out. Moreover, only the authority itself can give attributes with it as authority of the attribute.

At any time, a resource can verify if a requester can have access to its resource. It can check this information with `(11) authorizedByAttributeRules` function. It only needs to provide the requester and the resource identifiers. As this function does not change the state of the contract, it can be executed locally on the Ethereum node, generating no extra cost because its call does not get mined. In addition, for the same reason, it also has a similar response time from a common lightweight database.

Finally, in order to avoid centralization, we understand that the IoT access has to be the more direct as possible. Therefore, in E-ControlChain, the requester makes an attempt of `(10) Access` direct to the device, receiving an `(12) Acceptance or denial` to its attempt. However, as to `(5) Off-line authorization request`, the specification of the exactly way this off-line interactions occur are out of scope for the E-ControlChain.

Assumptions. Our assumptions are the following. (1) For each resource that requires a different access control policy, there is a different Ethereum network address and, therefore, a public and private key. (2) Only the resource device knows the private keys for its resources. (3) If the resource device does not have the minimum power to run a Ethereum client, it has a trusted third party node who can run the client for it. (4) The Ethereum client can synchronize correctly with the network, ie, there are no communication problems between the client node and the network. Furthermore, there are no attackers capable of blocking or changing the synchronization data. (5) A resource device can verify and extract the address of a requester or has a trusted third party that can do it. (6) No one, except the owner, can have access to or change the device private key. As can be seen, all the assumptions are reasonable taking into account the Blockchain mode of operation, its security mechanisms, and Ethereum characteristics.

5.1 | Experimental evaluation

We divided the experimental evaluation section in three parts. Firstly, we present our experimental environment (Section 5.1.1). Secondly, we expose the E-ControlChain usage cost (Section 5.1.2). Finally, we show the burden it causes on constrained resource devices (Section 5.1.3).

5.1.1 | Experimentation environment

For these evaluations, we used the environment depicted in Figure 8. We have a Raspberry Pi 3 B+ that has some simulated sensors and actuators plugged to it. It also runs an Ethereum client and uses a Decentralized Application (DApp) interface, implemented in JavaScript, to interact with E-ControlChain contract published on Ethereum. This Ethereum client does not mine; it only synchronizes with the Ethereum network.

The resource owners, authorities, and requesters also interact with the E-ControlChain contract on Ethereum network trough a DApp interface. Like the resource devices, with this interface, they can call functions of E-ControlChain to publish access rules, attributes, context information, and so on. The attempt of access resources can be done using IoT communication protocols, like MQTT or CoAP. Note that, for MQTT, a broker is required to intermediate the communication between requesters and resource devices. Therefore, the DApp interface and the Ethereum node client are required to be on the broker also. In our experimentation, we used the direct access.

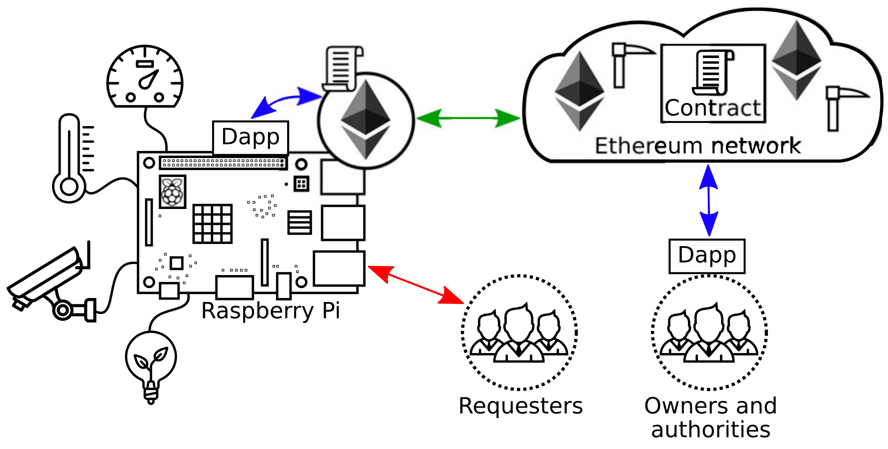


FIGURE 8 Experimentation environment

TABLE 5 Cost of executing the functions with a gas price of 20 gwei

Function	Cost in Wei	Cost in Ether	Cost in USD[b]
E-Controlchain contract deployment	$8.6 * 10^{16}$	0.08658	17.316
setOwner	$1.1 * 10^{15}$	0.00113	0.226
approveOwnership	$8.8 * 10^{14}$	0.00088	0.176
setRulesAuthority	$9.2 * 10^{14}$	0.00092	0.184
newContextRule	$2.3 * 10^{15}$	0.00229	0.458
createAttributeRule	$7.5 * 10^{15}$	0.00751	1.350
setAttribute	$9.2 * 10^{14}$	0.00092	0.184
setContext	$9.4 * 10^{14}$	0.00094	0.188
authorizedByAttributeRules[a]	0	0.00000	0.000

[a]Although it is executed over the contract, it is executed locally on the Ethereum client and does not generate extra cost. Only functions that change the state of the network are charged by the extra cost.
[b]Based on the price of ether from October 29, 2018 (around USD 200.00 for 1 ether).

Finally, the E-ControlChain contract for this experiment was implemented in Solidity version 0.4.23 and the version of running Ethereum clients and miners was 1.8.17 (the pseudocode is available in the Appendix). The Raspberry Pi was running Raspbian released on 2018-06-27 with kernel version 4.14. Furthermore, to emulate the Ethereum network, we used an Ubuntu 18.04 with kernel 4.15 over a MacBook Pro with core i7, 16 GB of RAM, and SSD storage.

5.1.2 | Usage cost

The approximated cost of executing each E-ControlChain function is presented on Table 5. Note that this cost can be drastically reduced adopting other mechanisms of "mining" instead the proof-of-work. For example, it could use the proof-of-authority, or approaches based on Directed Acyclic Graph (DAG) (like IoT Chain and IOTA[28,29]) or based on Byzantium algorithms (like the Algorand[27]).

5.1.3 | Burden on constrained devices

One important factor about an access control architecture for the IoT is its viability in scenarios with constrained resource devices. In this section, we show the results of running the E-ControlChain over the Raspberry Pi and discuss how ControlChain can be compatible with even more constrained devices.

In order to measure the impact of E-ControlChain over the Raspberry Pi, we collected its resource usage in three different scenarios. In the first one, "stand by," it was in a resting state, ie, only with its fundamental activities running (basic operational system tasks). In the second one, "geth," a Ethereum client (geth) was running, however with no contract interaction. Finally, in the third one, "geth + CD + CI", an Ethereum client was running with contract interactions in a way that it saturates the capacity of the Ethereum mining process. In order to do this, we generated 1000 transactions and, as soon as all of them were mined, we generated another 1000 transactions and so on. Using this methodology, we achieve an average of 28 mined transactions per second. Note that the Ethereum client in the Raspberry Pi only processes transactions from mined blocks that arrives in approximated intervals of 10 seconds.

The generated transactions was a mix of `setRulesAuthority`, `newContextRule`, `createAttributeRule`, `setAttribute`, and `setContext` functions. The `setOwner`, `approveOwnership`, and `authorizedByAttributeRules` functions were not used because the first two are security dependencies for the `setRulesAuthority` and the last is executed locally in the geth node and, consequently, are not mined into the Blockchain. The second and third scenarios were used to reveal how much of the burden caused by E-ControlChain is actually caused by the Ethereum fundamental activities. Finally, the measurements were obtained using the collectd v5.7.1 tool with the collection interval set to 5 seconds and all measurement graphs are an average of 10 repetitions.

The "geth" and "geth + CD + CI" scenarios can be divided in periods. The "geth" can be divided in two periods, ie, before geth complete initialization (BG) and after geth complete initialization (AG). The "geth + CD + CI" can be divided in three periods, ie, BG, contract deployment (CD), and contract interactions (CI). The approximated time in which the division between them occurs is represented in the result graphs as two vertical dashed lines. The first one divides the BG from AG and BG from CD. The second one divides the CD from CI.

Figure 9 presents the Raspberry Pi CPU and memory usage for the three scenarios. The CPU results (Figure 9A) shows that the E-ControlChain causes a CPU consumption peak of 35% in the BG and CI period. The result also shows that the Ethereum client is only responsible for 2% of the CPU usage after geth initialization and synchronization and that the operational system CPU usage in stand by is around 0.1%. On the other

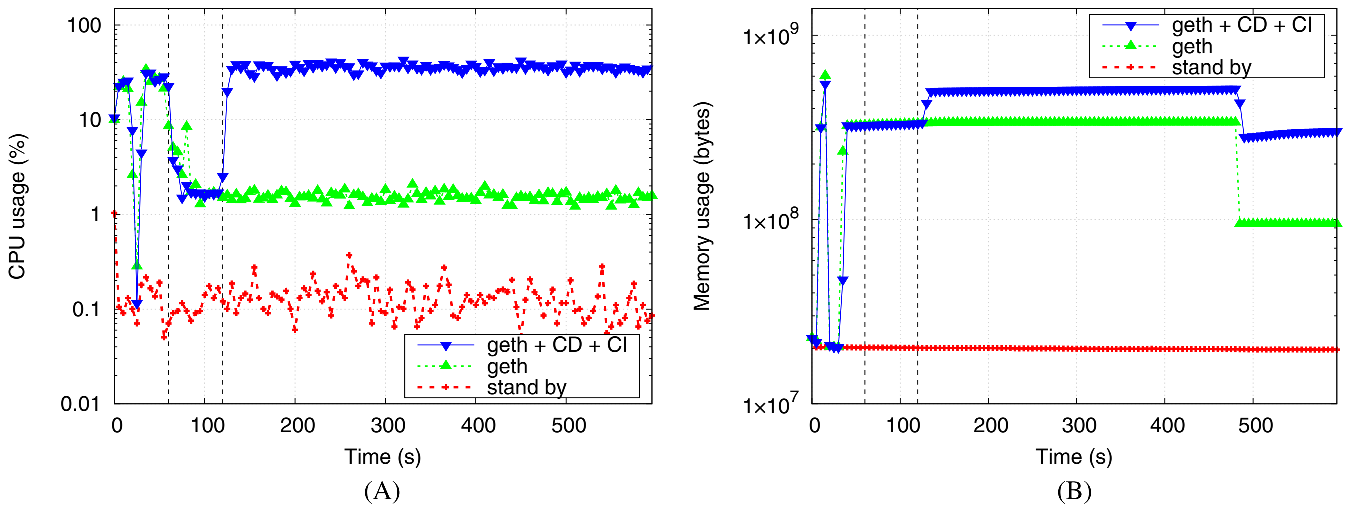


FIGURE 9 CPU and memory impact. A, CPU; B, Memory

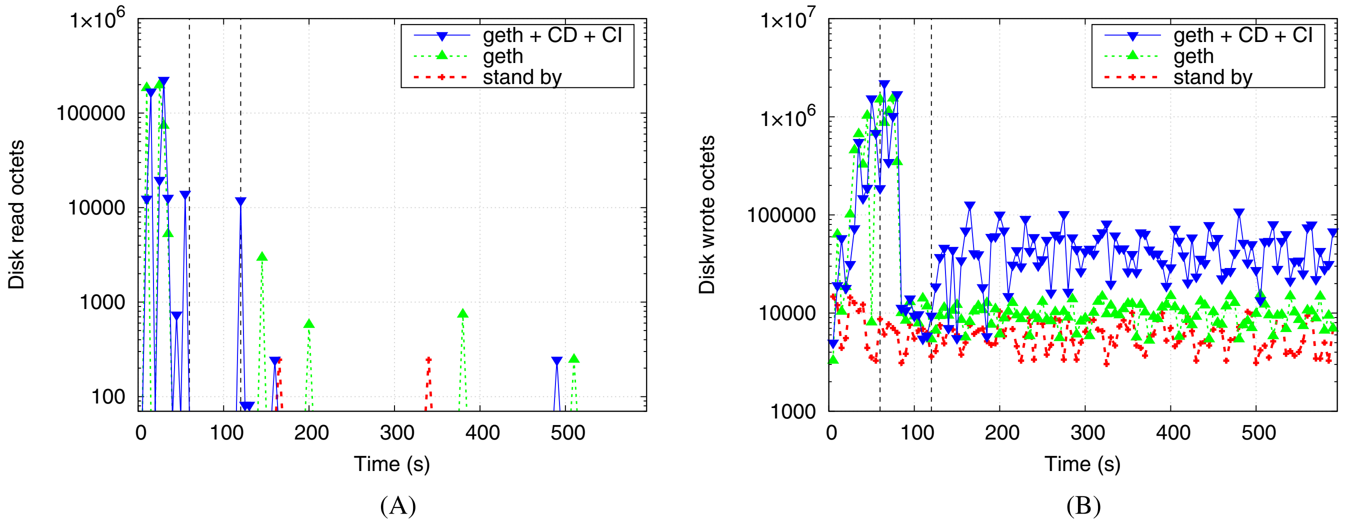
hand, the memory usage results (Figure 9B) shows the same order of magnitude on geth scenarios (350 MB versus 500 MB in the CI period). Their memory usage drop around the 485 seconds could be the release of memory used to store unused geth code or information that after a while in inactivity gets freed; however, a more deep investigation is necessary to reveal the real reasons behind it. In the stand by scenario, the memory usage was around 20 MB.

The measurements obtained from disk reading, writing, and total usage are presented in Figure 10. The disk read measurements (Figure 10A) shows very few readings in the AG, CD, and CI periods with the more overloading activity being the reading of 10 KB in an interval of 5 seconds. In the BG period, there were peaks of 200 KB. In counterpart, the write measurements (Figure 10B) show a slight increase in "geth" scenario when compared to "stand by" one. The "geth + CD + CI" scenario shows peaks 10x higher than the "geth scenario"; however, these peaks are about 100 KB. In the BG period, there are write peaks of 2 MB. The stored information on disk (Figure 10C) shows an increase of 30 MB in the BG period and a linear use of the storage space in CI period. In almost 500 seconds of contract interactions in the CI period, it used approximately 16 MB. This measures up to 2.7 GB of storage requirement per day. If a real E-ControlChain was heavily used as in this experimentation, perhaps this could be the main problem of using it over the IoT network. Fortunately, as discussed after, there are some ways to reduce or avoid this problem.

Figure 11 presents the collected values of the network interface. Figures 11A and 11B show the quantity of received and transmitted octets, respectively. As expected, there was an increase in the network traffic, which is necessary to keep the network synchronized. The "geth" scenario received and transmitted, in average, 1 KB and 750 B, respectively, for each interval of 5 seconds. The "geth + CD + CI" scenario received and transmitted a bigger quantity of information in the CI period (received 30 KB and transmitted 10 KB of information approximately for each interval of 5 seconds). The "stand by" scenario keeps the information transmission and receipt lower than 20 bytes for each interval of 5 seconds.

The results obtained in these experiments show the viability of running E-ControlChain in even more resource constrained devices than the used one. However, even so, for devices with a severe storage restriction, it can be applied a replication factor to the stored data, ie, each device keeps a part of the data with a probability of *x*%, instead of it in totality. It is possible to apply selectively information storage also, making them keep only information related to themselves and, of course, the ones necessary to keep updating with the E-ControlChain (like some of the most recent mined blocks). Finally, supporting devices can be used for helping those even more constrained devices (that cannot handle CPU, memory, or network overload imposed by E-ControlChain). Therefore, when they receive an access request, they forward it to a supporting device that will check the authorization for them. Of course, the process of selecting these supporting devices requires caution; otherwise, malicious ones can end up being selected, compromising the device depending on it. We suggest that, when possible, the constrained device always select devices owned by its own owner or from its confidence circle.

To demonstrate the potential of a single Raspberry Pi as a support device for other devices (ie, that checks authorization in behalf of other devices and reduce overload on them), we made a stress authorization test over three scenarios (Figure 12). In the first one, "Without CI," the E-ControlChain was deployed, but there were no transactions towards it. In the second one, "During CI," the Raspberry Pi was under the scenario described for CI period, where the Ethereum network was saturated with transactions towards the contract. In the third one, there were no contract transactions; however, it occurred after 600 seconds of CI period. The results show that the number of transactions made to the contract has no or little influence over the number of authorizations the Raspberry can check. Both, "Without CI" and "After 600s of CI" scenarios presented a similar result of approximately 30 authorization per second. However, in "During CI" scenario, the average of authorizations dropped to approximately 27, showing a reduction of about 10% with the intense transaction mining. This reduction is, probably, caused by mechanisms that deal with the concurrency of reading from and writing to the contract at the same time. Furthermore, for big networks, multiple Raspberry Pi devices can be used, and is expected that the number of authorizations per second will be arithmetically expanded.



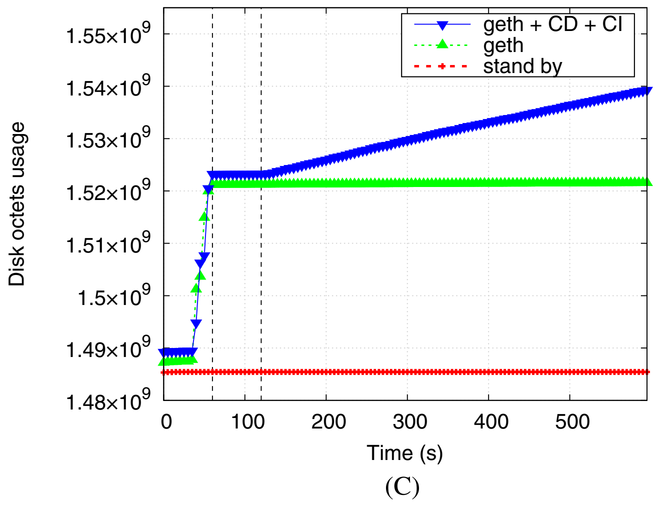


FIGURE 10 Disk impact. A, Read; B, Write; C, Usage

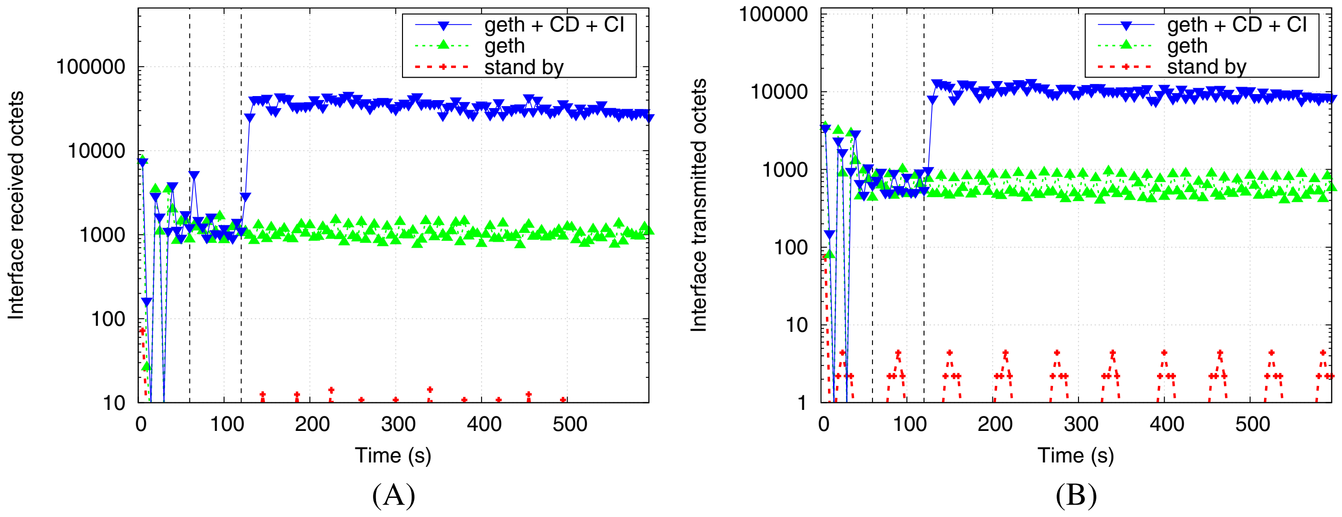


FIGURE 11 Network receive and transmission impact. A, Network receiving impact; B, Network transmission impact

5.2 | Limitations

The limitations of E-ControlChain can be divided in three main aspects, namely, design decisions, base components, and algorithm decisions. The first aspect, design decision, are limitations that involve the basic concepts behind E-ControlChain. The base components are limitations imposed

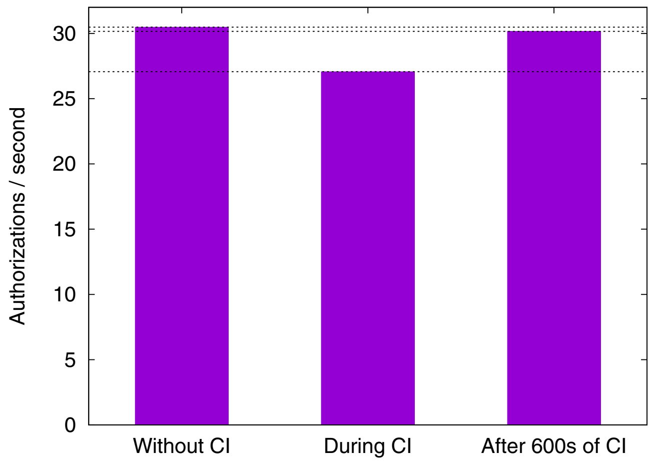


FIGURE 12 Number of authorizations per second

by the adopted platforms, libraries, or other software components used in the development. Finally, the algorithm decisions are limitations that was taken in the development stage in favor of other benefits.

The first limitation caused by design decisions was the choosing of a proof-of-work-based Blockchain as a central tool for E-ControlChain. Blockchains based on proof-of-work are at risk of miners collusion. Although it does not give unauthorized access to devices because of the application security policies, it can prevent some users to manage its own devices and addresses. As much as it is a possible scenario, at least two things make it very unlikely to happen. First, it would require more than 50% of the mining power to be in the control of the colluding miners. Second, even if they succeed in the control of such mining power, they are going to commit suicide as the network will loss reputation and people will start to abandon it and, therefore, miners will lose their source of income. Thus, it is expected that the financial disadvantages prevent miners collusion.

Another limitation generated by design decisions is the adoption of a public Blockchain and the implementation of all algorithm decisions in the contract. The drawback of this approach is that it lacks of a strong privacy protection. Although, all the devices and users are behind addresses, which are known to provide a certain level of anonymity, profiling tools can be applied and, in some cases, could end up revealing the real identity behind them.[77] As we chose to implement all the algorithm decisions in the contract, most of the stored information cannot be really encrypted. However, some data can be replaced by another one using a bijective function and, therefore, making it difficult the deduction of what the data represents and what the real values are behind the masked data. Furthermore, in future works, we are planning to create a version of E-ControlChain over a private Blockchain and other more privacy-friendly Blockchain solutions.

The dependency on the public key pair is another limitation of design decisions. The loss of a key requires the generation of a new one and the configuration of it in E-ControlChain (definition of owner, attributes, access rules, and so on). Moreover, if an attacker has access to the private key, it can hijack the device, create fake messages in name of the device, and control other devices that are owned by it. However, in our design, the physical access prevails over any other control and allows, eg, the reconfiguration and change of new key pairs for the device to avoid new unwanted accesses or operations. This only requires the configuration of it on E-ControlChain. Unfortunately, this also has the drawback of allowing the usage of stolen devices.

We chose Ethereum as a platform and Solidity as a programming language for the first version of E-ControlChain. Naturally, each platform and language has its own limitations (whether intentional or not) that are forwarded to solutions that adopt them. In this sense, the E-ControlChain also inherited some of these basic components limitations. Probably, the most explicit of them is the requirement of an Ethereum client on the device itself or on its selected supporting device. Currently, there are compiled version of geth (one of the official implementations of the Ethereum clients) for Android, iOS, macOS, Windows, and for Linux using the architectures 32-bit, 64-bit, ARM64, ARMv5, ARMv6, ARMv7, MIPS32, and MIPS64. Thus, out-of-the-box, a wide range of devices is supported. The geth source code is also available and can facilitate its provision for those that are not supported yet.

The base component Solidity also brought some limitations to E-ControlChain. With the objective of keeping the contracts clean and avoid code bugs, solidity defines a limit to the number of variables defined in each function. In addition, the only way to pass/return objects to/from functions in the used version of the Solidity compiler is using the ABIEncoderV2; however, when we were making the tests, it was a little unstable and we decided not to use it. Therefore, the lack of a suitable solution to work with objects in the experimenting time worsened the “problem” of the limitation imposed to the number of variables in a function. Therefore, implementing more complex and complete access controlling became a harder task. Because of this, the current E-ControlChain does not implement obligations and accountabilities instructions. Furthermore, the owner can define only one context set for each rule and cannot create more complex allowed contexts, eg, using logical operators. Although there is no mystery in the implementation of these functions, it became a future work that could come true with a stable ABIEncoderV2.

Finally, there are also limitations coming from implementation decisions. Two of them are as follows: (1) only one rule for each pair resource-requester in the capability and ACL access control can be defined; (2) in order to remove an attribute-based rule, the context indexes and the allowed actions have to be passed in the same order they were passed in the rule creation. All these limitations could be avoided with more code instructions; however, this would turn the code more complex, difficult to inspect, and would increase its deployment and execution cost. Therefore, we choose to keep these limitations in the current version of E-ControlChain.

6 | CONCLUSION

The great services that the IoT can provide came together with many concerns. As the IoT will handle also private and confidential information, some of the concerns are about privacy and security of this information. We are experiencing an increasing wave of successful attacks targeting the IoT environment and devices. They only reveal to us the vulnerability and inefficiency of the currently adopted access controls systems. In this work, we have joined the on going effort to change this scenario and we have proposed a new access control architecture for the IoT, the ControlChain, and demonstrated its viability through a proof of concept, the E-ControlChain.

ControlChain is an architecture that believes in the power of the Blockchain technology. It was initially designed to support at least three types of authorization (based on attributes, capabilities, and ACL). Together, they are capable of providing compatibility with a wide range of already adopted IoT access control models and, therefore, can smooth the migration from older and outdated access control mechanisms. Furthermore, our architecture also includes a secure way of creating relationships, assigning attributes for them, and using them in the access control.

The main idea behind the ControlChain architecture is the provision of basic pillars for a simple and, at the same time, powerful access control system. It was based on the most popular IoT requirements, like scalability, transparency, fault-tolerance, compatibility, and many others. It gives the IoT devices the freedom for achieving its tasks efficiently, without letting the users lose the control over its owned devices and information.

The proof-of-concept of ControlChain, E-ControlChain, was implemented to run over the Ethereum platform. The results showed us that even limited devices, like a Raspberry Pi, can easily handle the E-ControlChain requirements. Furthermore, we also pointed directions on how more severe limited devices can also benefit from the E-ControlChain using more powerful and trusted devices as support.

ORCID

Otto Julio Ahlert Pinno https://orcid.org/0000-0001-9460-4750

REFERENCES

1. Gartner. Gartner IT Glossary: Internet of Things. 2017. http://www.gartner.com/it-glossary/internet-of-things/. Accessed April 25, 2017.
2. Nordrum A. Popular Internet of Things forecast of 50 billion devices by 2020 is outdated. 2016. http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated. Accessed March 16, 2017.
3. Manyika J, Chui M. By 2025, Internet of Things applications could have $11 trillion impact. 2015. https://www.mckinsey.com/mgi/overview/in-the-news/by-2025-internet-of-things-applications-could-have-11-trillion-impact. Accessed August 23, 2018.
4. Ouaddah A, Elkalam AA, Ouahman AA. FairAccess: a new blockchain-based access control framework for the Internet of Things. *Secur Commun Netw*. 2017;9(18):5943-5964.
5. Ouaddah A, Mousannif H, Elkalam AA, Ouahman AA. Access control in the Internet of Things: big challenges and new opportunities. *Comput Netw*. 2017;112:237-262.
6. Varadharajan V, Bansal S. Data security and privacy in the Internet of Things (IoT) environment. In: *Connectivity Frameworks for Smart Devices: The Internet of Things from a Distributed Computing Perspective*. Cham, Switzerland: Springer International Publishing; 2016;261-281.
7. Greenberg A. Hackers remotely kill a jeep on the highway-with me in it. 2015. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. Accessed August 24, 2017.
8. Greenberg A, Zetter K. How the Internet of Things got hacked. 2015. https://www.wired.com/2015/12/2015-the-year-the-internet-of-things-got-hacked/. Accessed August 24, 2017.
9. Krebs B. Hacked cameras, DVRs powered today's massive internet outage. 2016. https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/. Accessed November 3, 2016.
10. Symantec Security Response. VPNFilter: new router malware with destructive capabilities. 2018. https://www.symantec.com/blogs/threat-intelligence/vpnfilter-iot-malware. Accessed June 13, 2018.
11. Sandhu RS, Samarati P. Access control: principle and practice. *IEEE Commun Mag*. 1994;32(9):40-48.
12. Zhou Y, Zhang Y, Fang Y. Access control in wireless sensor networks. *Ad Hoc Netw*. 2007;5(1):3-13.
13. Seitz L, Selander G, Gehrmann C. Authorization framework for the Internet-of-Things. Paper presented at: 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM); 2013; Madrid, Spain.
14. Ouaddah A, Elkalam AA, Ouahman AA. 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*. Cham, Switzerland: Springer International Publishing AG; 2017:523-533.
15. bitcoin. Bitcoin: a peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf. Accessed April 11, 2017.
16. Swan M. *Blockchain: blueprint for a new economy*. Sebastopol, CA: O'Reilly Media; 2015.
17. Buterin V. A next-generation smart contract and decentralized application platform. Ethereum Project; 2017. https://github.com/ethereum/wiki/wiki/white-paper. Accessed March 14, 2017.

18. Georgiev G. South Korea to invest $230 M in blockchain technology development. 2018. https://bitcoinist.com/south-korea-230b-invest-blockchain. Accessed June 25, 2018.

19. Pinno OJA, Gregio ARA, De Bona LCE. ControlChain: blockchain as a central enabler for access control authorizations in the IoT. Paper presented at: 2017 IEEE Global Communications Conference (GLOBECOM); 2017; Singapore.

20. Bai G, Yan L, Gu L, Guo Y, Chen X. Context-aware usage control for web of things. *Secur Commun Netw*. 2014;7(12):2696-2712.

21. Liu Y, Zhang Y, Ling J, Liu Z. Secure and fine-grained access control on e-healthcare records in mobile cloud computing. *Futur Gener Comput Syst*. 2018;78(3):1020-1026.

22. Yang K, Han Q, Li H, Zheng K, Su Z, Shen X. An efficient and fine-grained big data access control scheme with privacy-preserving policy. *IEEE Internet Things J*. 2017;4(2):563-571.

23. Zhou Q, Elbadry M, Ye F, Yang Y. Flexible, fine grained access control for Internet of Things: Poster abstract. In: Proceedings of the Second International Conference on Internet-of-Things Design and Implementation (IoTDI); 2017; Pittsburgh, PA.

24. Consalvo M. Using your friends: social mechanics in social games. In: Proceedings of the 6th International Conference on Foundations of Digital Games (FDG); 2011; Bordeaux, France.

25. Yli-Huumo J, Ko D, Choi S, Park S, Smolander K. Where is current research on blockchain technology?–A systematic review. *PLoS One*. 2016;11(10):1-27.

26. Lin I-C, Liao T-C. A survey of blockchain security issues and challenges. *Int J Netw Secur*. 2017;19(5):653-659.

27. Gilad Y, Hemo R, Micali S, Vlachos G, Zeldovich N. Algorand: scaling Byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles (SOSP); 2017; Shanghai, China.

28. IoT Chain. *IoT Chain: A High-Security Lite IoT OS*. White paper. 2018. https://iotchain.io/whitepaper/ITCWHITEPAPER.pdf. Accessed September 9, 2018.

29. IOTA. What is IOTA? A permissionless distributed ledger for a new economy. 2018. https://www.iota.org/get-started/what-is-iota. Accessed September 9, 2018.

30. Wood G. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Ethereum project yellow paper. 2014.

31. Etherscan: The Ethereum Block Explorer. Unique address growth chart. https://etherscan.io/chart/address. Accessed July 26, 2018.

32. Wilmoth J. More than 1,000 Ethereum DApps have launched since 2017. 2018. https://www.ccn.com/more-than-1000-ethereum-dapps-have-launched-since-2017/. Accessed June 25, 2018.

33. Ouaddah A, Elkalam AA, Ouahman AA. Harnessing the power of blockchain technology to solve IoT security & privacy issues. In: Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing (ICC); 2017; Cambridge, UK.

34. Zhang Y, Kasahara S, Shen Y, Jiang X, Wan J. Smart contract-based access control for the Internet of Things. *IEEE Internet of Things J*. 2018:1-1.

35. Kumar NM, Mallick PK. Blockchain technology for security issues and challenges in IoT. *Procedia Comput Sci*. 2018;132:1815-1823. Part of Special Issue: International Conference on Computational Intelligence and Data Science.

36. Nuss M, Puchta A, Kunz M. Towards blockchain-based identity and access management for Internet of Things in enterprises. In: Furnell S, Mouratidis H, Pernul G, eds. *Trust, Privacy and Security in Digital Business: 15th International Conference, TrustBus 2018, Regensburg, Germany, September 5-6, 2018, Proceedings*. Cham, Switzerland: Springer Nature Switzerland AG; 2018:167-181.

37. Panarello A, Tapas N, Merlino G, Longo F, Puliafito A. Blockchain and IoT integration: a systematic survey. *Sensors*. 2018;18(8):2575.

38. Azaria A, Ekblaw A, Vieira T, Lippman A. MedRec: using blockchain for medical data access and permission management. Paper presented at: 2016 2nd International Conference on Open and Big Data (OBD); 2016; Vienna, Austria.

39. Fujimura S, Watanabe H, Nakadaira A, Yamada T, Akutsu A, Kishigami JJ. BRIGHT: a concept for a decentralized rights management system based on blockchain. Paper presented at: 2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin); 2015; Berlin, Germany.

40. Zyskind G, Nathan O, Pentland A. Decentralizing privacy: using blockchain to protect personal data. Paper presented at: 2015 IEEE Security and Privacy Workshops; 2015; San Jose, CA.

41. IBM. Watson Internet of Things: the Internet of Things becomes the internet that thinks with Watson IoT. 2017. https://www.ibm.com/internet-of-things/. Accessed August 13, 2017.

42. Dukkipati C, Zhang Y, Cheng LC. Decentralized, blockchain based access control framework for the heterogeneous Internet of Things. In: Proceedings of the Third ACM Workshop on Attribute-Based Access Control (ABAC); 2018; Tempe, AZ.

43. Zhu Y, Qin Y, Zhou Z, Song X, Liu G, Chu WC. Digital asset management with distributed permission over blockchain and attribute-based access control. Paper presented at: 2018 IEEE International Conference on Services Computing (SCC); 2018; San Francisco, CA.

44. Xu R, Chen Y, Blasch E, Chen G. A federated capability-based access control mechanism for Internet of Things (IoTs). CoRR. 2018. https://arxiv.org/abs/1805.00825

45. Xu R, Chen Y, Blasch E, Chen G. BlendCAC: a blockchain-enabled decentralized capability-based access control for IoTs. CoRR. 2018. https://arxiv.org/abs/1804.09267

46. Pohrmen FH, Das RK, Khongbuh W, Saha G. Blockchain-based security aspects in Internet of Things network. In: Luhach AK, Singh D, Hsiung P-A, Hawari KBG, Lingras P, Singh PK, eds. *Advanced Informatics for Computing Research: Second International Conference, ICAICR 2018, Shimla, India, July 14-15, 2018, Revised Selected Papers, Part II*. Singapore: Springer Nature Singapore Pte Ltd; 2019:346-357.

47. Klaokliang N, Teawtim P, Aimtongkham P, So-In C, Niruntasukrat A. A novel IoT authorization architecture on hyperledger fabric with optimal consensus using genetic algorithm. Paper presented at: 2018 Seventh ICT International Student Project Conference (ICT-ISPC); 2018; Nakhonpathom, Thailand.

48. Linux Foundation. Hyperledger fabric. 2019. https://www.hyperledger.org/projects/fabric. Accessed January 12, 2019.

49. Wu L, Du X, Wang W, Lin B. An out-of-band authentication scheme for Internet of Things using blockchain technology. Paper presented at: 2018 International Conference on Computing, Networking and Communications (ICNC); 2018; Maui, HI.

50. Hammi MT, Hammi B, Bellot P, Serhrouchni A. Bubbles of trust: a decentralized blockchain-based authentication system for IoT. *Comput Secur*. 2018;78:126-142.

51. Lin C, He D, Huang X, Choo KKR, Vasilakos AV. BSeIn: a blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *J Netw Comput Appl*. 2018;116:42-52.

52. Molina-Jimenez C, Solaiman E, Sfyrakis I, Ng I, Crowcroft J. On and off-blockchain enforcement of smart contracts. ArXiv e-prints. 2018.

53. Casey MJ. 'Layer 2' blockchain tech is an even bigger deal than you think. 2018. https://www.coindesk.com/layer-2-blockchain-tech-even-bigger-deal-think. Accessed June 25, 2018.

54. Abd El-Aziz AA, Kannan A. A comprehensive presentation to XACML. Paper presented at: Third International Conference on Computational Intelligence and Information Technology (CIIT); 2013; Mumbai, India.

55. OASIS. eXtensible access control markup language (XACML) version 3.0. 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. Accessed May 1, 2017.

56. Hardt D. RFC 6749: the OAuth 2.0 authorization framework. RFC Editor; 2012.

57. Kantara Initiative Inc. User-managed Access (UMA). 2017. https://kantarainitiative.org/confluence/display/uma/home. Accessed April 5, 2017.

58. Barka E, Mathew SS, Atif Y. Securing the Web of Things with Role-based access control. In: *Codes, Cryptology, and Information Security: First International Conference, C2SI 2015, Rabat, Morocco, May 26-28, 2015, Proceedings - In Honor of Thierry Berger*. Cham, Switzerland: Springer International Publishing; 2015;14-26.

59. Jindou J, Xiaofeng Q, Cheng C. Access control method for Web of Things based on role and SNS. Paper presented at: 2012 IEEE 12th International Conference on Computer and Information Technology; 2012; Chengdu, China.

60. Liu J, Xiao Y, Chen CLP. Authentication and access control in the Internet of Things. Paper presented at: 2012 32nd International Conference on Distributed Computing Systems Workshops; 2012; Macau, China.

61. Zhang G, Tian J. An extended role based access control model for the Internet of Things. Paper presented at: 2010 International Conference on Information Networking and Automation (ICINA); 2010; Kunming, China.

62. Ye N, Zhu Y, Wang R, Malekian R, Qiao-min L. An efficient authentication and access control scheme for perception layer of Internet of Things. *Appl Math Inf Sci.* 2014;8(4):1617-1624.

63. Zhang G, Gong W. The research of access control based on UCON in the Internet of Things. *J Softw.* 2011;6(4):724-731.

64. Hernández-Ramos JL, Jara AJ, Marín M, Skarmeta Gómez AF. DCapBAC: embedding authorization logic into smart things through ECC optimizations. *Int J Comput Math.* 2016;93(2):345-366.

65. Mahalle PN, Anggorojati B, Prasad NR, Prasad R. Identity authentication and capability based access control (IACAC) for the Internet of Things. *J Cyber Secur Mobil.* 2013;1(4):309-348.

66. Anggorojati B, Mahalle PN, Prasad NR, Prasad R. Secure access control and authority delegation based on capability and context awareness for federated IoT. In: *Internet of Things and M2M Communications.* Aalborg, Denmark: River Publishers; 2013:135-160.

67. Gusmeroli S, Piccione S, Rotondi D. A capability-based security approach to manage access control in the Internet of Things. *Math Comput Model.* 2013;58(5-6):1189-1205. Part of Special Issue: The Measurement of Undesirable Outputs: Models Development and Empirical Analyses and Advances in mobile, ubiquitous and cognitive computing.

68. Bernabe JB, Ramos JLH, Skarmeta Gomez AF. TACIot: multidimensional trust-aware access control system for the Internet of Things. *Soft Comput.* 2016;20(5):1763-1779.

69. Neisse R, Fovino IN, Baldini G, Stavroulaki V, Vlacheas P, Giaffreda R. A model-based security toolkit for the Internet of Things. Paper presented at: 2014 Ninth International Conference on Availability, Reliability and Security; 2014; Fribourg, Switzerland.

70. Mahalle PN, Thakre PA, Prasad NR, Prasad R. A fuzzy approach to trust based access control in Internet of Things. Paper presented at: Wireless VITAE 2013; 2013; Atlantic City, NJ.

71. Ferraiolo D, Kuhn R. Role-based access control. Paper presented at: 15th NIST-NCSC National Computer Security Conference; 1992; Baltimore, MD.

72. Kalam AAE, Baida RE, Balbiani P, et al. Organization based access control. In: Proceedings POLICY 2003 IEEE 4th International Workshop on Policies for Distributed Systems and Networks; 2003; Lake Como, Italy.

73. Hu VC, Ferraiolo D, Kuhn R, et al. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations.* Technical report. Gaithersburg, MD: National Institute of Standards and Technology; 2014.

74. Park J, Sandhu R. The UCONABC usage control model. *ACM Trans Inf Syst Secur.* 2004;7(1):128-174.

75. Anggorojati B, Prasad NR, Prasad R. Secure capability-based access control in the M2M local cloud platform. Paper presented at: 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace and Electronic Systems (VITAE); 2014; Aalborg, Denmark.

76. Solidity. Introduction to smart contracts. 2018. http://solidity.readthedocs.io/en/latest/introduction-to-smart-contracts.html. Accessed June 13, 2018.

77. Huang B, Liu Z, Chen J, Liu A, Qi L, He Q. Behavior pattern clustering in blockchain networks. *Multimed Tools Appl.* 2017;76(19):20099-20110.

How to cite this article: Pinno OJA, Grégio ARA, De Bona LCE. ControlChain: A new stage on the IoT access control authorization. *Concurrency Computat Pract Exper*. 2019;e5238. https://doi.org/10.1002/cpe.5238

APPENDIX

Next, we expose, as a pseudocode, the crucial parts of the E-ControlChain attribute-based authorization. Listings A1 and A2 show the pseudocode of the basic and common control, and the attribute-based access control of E-ControlChain, respectively. The "BasicAndCommonControl" defines variables and functions that are common to all the three types of rules defined by the ControlChain, namely, attribute, capability, and ACL. The "AttributeControl" defines the variables and functions that are particularly related to the attribute-based access control. In addition, note that the "AttributeControl" inherits the "BasicAndCommonControl" and, thus, all its variables and functions.

```
Contract BasicAndCommonControl {
        address[address] owners;
        bool[address] approved_ownerships;
        address[address] rules_authorities;
        int[address][string] context;
        ContextRule[uint] context_rules;
        struct ContextRule {
                address source;
                string identifier;
                Comparators comparator;
                int value;
        }
        function setOwner(address addr, address new_owner) {
                REQUIREMENT: be the entity behind "addr" or the current owner of it
                approved_ownerships[addr] = false;
                owners[addr] = new_owner;
                rules_authorities[addr] = 0x00;
        }
        function approveOwnership(address addr) {
                REQUIREMENT: be the new owner of "addr"
                approved_ownerships[addr] = true;
        }
        Function setRulesAuthority(address resource, address addr) {
                REQUIREMENT: be the "resource" itlsef or its current owner
                REQUIREMENT 2: "resource" ownership is with approved status
                rules_authorities[resource] = addr;
        }
        function setContext(address source, string identifier, int value) {
                REQUIREMENT: be the entity behind "addr" or the current owner of it
                context[source][identifier] = value;
        }
        function newContextRule(address source, string identifier, Comparators comparator, int value){
                ContextRule context_rule = ContextRule(source, identifier, comparator, value);
                context_rules.push(context_rule);
        }
        function isContextSatisfied(uint[uint] context_rules_ indexes) {
                foreach (i in context_rules_indexes) {
                        if (context_rule[i] doesn't hold) {
                                return false;
                        }
                }
                return true;
        }
}
```

Listing A1 E-ControlChain - BasicAndCommonControl pseudocode

```
Contract AttributeControl extends BasicAndCommonControl {
        bool[address][address][string] attributes;
        AttributeRule[address][uint] attribute_rules;
        Struct AttributeRule {
                address resource_authority;
                string resource_attr;
                address requester_authority;
                string requester_attr;
                bool inherit_owner_attr;
                uint[uint] context;
                Actions[uint] actions;
        }
        Function setAttribute(address addr, string attr, bool assigned) {
                attributes[msg.sender][addr][attr] = assigned;
        }
        Function createAttributeRule(address resource_authority, string resource_attr, address
            requester_authority, string requester_attr, bool inherit_owner_attr, uint[uint] context,
            Actions[uint] actions) {
                AttributeRule attribute_rule = AttributeRule(resource_authority, resource_attr,
                    requester_authority, requester_attr, inherit_owner_attr, context, actions);
                attribute_rules[msg.sender].push(attribute_rule);
        }
        Function authorizedByAttributeRules(address resource, address requester, Actions action) {
                foreach (ar in attribute_rules[rules_authorities[resource]]) {
                        if (the ar.resource_authority defined the ar.resource_attr for the resource) {
                                if (the ar.requester_authority defined the ar.requester_attr for the
                                    requester or (the inherit of requester owner attributes by the
                                    requester is allowed by the ar and the ownership of the resource
                                    was approved and the ar.requester_authority defined the ar.
                                    requester_attr for the owner of the requester)) {
                                        if{isContextSatisfied(ar.context) \textbf{and} ar.actions
                                            contains action} {
                                                return true;
                                        }
                                }
                        }
                }
                return false;
        }
}
```

Listing A2 E-ControlChain - AttributeControl pseudocode

Although, in Ethereum, it works more similar to a hash table with infinite entries and where we cannot iterate over the hash keys, when the pseudocode shows a variable type declaration as, eg, ‘‘bool[address],’’ it can be understood as an array of bool (Booleans) that requires an index of type ‘‘address,’’ and when its declaration is ‘‘someType[uint],’’ it is an array of ‘‘someType’’ that requires an uint (unsigned int) index.