Article

BlendCAC: A Smart Contract Enabled Decentralized Capability-based Access Control Mechanism for IoT

Ronghua Xu[1], Yu Chen[1]*, Erik Blasch [2], Genshe Chen[3]

[1] Binghamton University, SUNY, Binghamotn, NY 13902, USA; rxu22@binghamton.edu
[2] The U.S. Air Force Research Lab, Rome, NY 13441, USA; erik.blasch@gmail.com
[3] Intelligent Fusion Technology, Inc. Germantown, MD 20876, USA; gchen@intfusiontech.com
* Correspondence: ycheng@binghamton.edu; Tel.: +1-607-777-6133

Abstract: While the Internet of Things (IoT) technology has been widely recognized as the essential part of Smart Cities, it also brings new challenges in terms of privacy and security. Access control (AC) is among the top security concerns, which is critical in resource and information protection over IoT devices. Traditional access control approaches, like Access Control Lists (ACL), Role-based Access Control (RBAC) and Attribute-based Access Control (ABAC), are not able to provide a scalable, manageable and efficient mechanism to meet the requirements of IoT systems. Another weakness in today's AC is the centralized authorization server, which can be the performance bottleneck or the single point of failure. Inspired by the smart contract on top of a blockchain protocol, this paper proposes BlendCAC, which is a decentralized, federated capability-based AC mechanism to enable an effective protection for devices, services and information in large scale IoT systems. A federated capability-based delegation model (FCDM) is introduced to support hierarchical and multi-hop delegation. The mechanism for delegate authorization and revocation is explored. A robust identity-based capability token management strategy is proposed, which takes advantage of the smart contract for registering, propagating and revocating of the access authorization. A proof-of-concept prototype has been implemented on both resources-constrained devices (*i.e.*, Raspberry PI node) and more powerful computing devices (*i.e.*, laptops), and tested on a local private blockchain network. The experimental results demonstrate the feasibility of the BlendCAC to offer a decentralized, scalable, lightweight and fine-grained AC solution for IoT systems.

Keywords: Decentralized Access Control; Internet of Things (IoT); Blockchain Protocol; Smart Contract; Federated Delegation; Capability-based Access Control.

1. Introduction

With the proliferation of the Internet of Things (IoT), a large number of physical devices are being connected to the Internet at an unprecedented scale. The prevalence of the IoT devices changes human activities by ubiquitously providing applications and services that are revolutionizing transportation, healthcare, industrial automation, emergency response, and so on [1]. These capabilities offer both measurement data and information context for situation awareness (SAW) [2,3]. While benefiting from the large-scale applications like Smart Gird and Smart Cities, the quick growing IoT systems also incur new concerns for security and privacy. With the increased popularity, the connected smart IoT devices without sufficient security measures increase the risk of privacy breaches and various attacks. Security issues, such as privacy, authentication, access control, system configuration, information storage and management, are the main challenges that these IoTs based applications are facing [4].

Among the top security challenges in IoT environments, access authorization is critical in resource and information protection. Conventional access control approaches, like Access Control List (ACL), Role-based Access Control (RBAC) and Attribute-based Access Control (ABAC) have been widely used on information technology (IT) system. However, they are not able to provide a manageable and efficient mechanism to meet the requirements raised by IoT networks:

- *Scalability*: The fast growing number of devices and services also pose increasing management overload in access control systems that are based on ACL or RBAC models. Access control strategies are expected to be able to handle the scalability problem resulting from the distributed IoT networks.
- *Heterogeneity*: IoT systems normally integrate heterogeneous cyber physical objects with variant underlying technologies or in different application domains, and each domain or platform has its own specific requirements for identity authentication and authorization policy enforcement. Both RBAC and ABAC have been found inflexible to provide complex arrangements to support delegation and transitivity, which are essential for efficient and effective intra-domain authorization and access control.
- *Causality*: Traditional RBAC and ABAC systems envisage planned and long-lived patterns, while the IoT world is mainly characterized by short-lived, often causal and/or spontaneous interactions [5], in which an access control scheme is required to deal with dynamic challenges.
- *Lightweight*: IoT devices are usually resource-constrained, which cannot support heavy computational and large storage required applications, and those smart devices connect to each other by low power and lossy networks. Consequently the access control protocol should be lightweight and not impose significant overhead on devices and communication networks.

The extraordinary large number of devices with heterogeneity and dynamicity necessitate more scalable, flexible and lightweight access control mechanisms for IoT networks. In addition, a majority of the AC solutions rely on centralized authorities. Although the delegation mechanism helps migrate certain intelligence from the centralized cloud server to a near-site fog or edge network, the power of policy decision making and identity management is exclusively located in the cloud center. IoT networks need a new AC framework that provides decentralized authentication and authorization scheme in trustless application network environments, such that intelligence could be diffused among large number of distributed edge devices.

While being well-known as the fundamental protocol of Bitcoin [6], the first digital currency, the blockchain protocol has been recognized as the potential to revolutionize the fundamentals of IT technology because of its many attractive features and characteristics such as supporting decentralization and anonymity maintenance [7]. In this paper, a *BLockchain-ENabled, Decentralized, Federated, Capability-based Access Control* (BlendCAC) scheme is proposed to enhance the security of IoT devices. It provides a decentralized, scalable, fine-grained, and lightweight AC solution to protect smart devices, services and information in IoT networks. An identity-based capability token management strategy is presented and the federated authorization delegation mechanism is illustrated. In addition, a capability-based access validation process is implemented on service providers that integrate SAW and customized contextualized conditions. The experimental results demonstrate the feasibility and effectiveness of the proposed BlendCAC scheme.

The major contributions of this work are:

1. Leveraging the blockchain protocol, a decentralized, federate access control scheme is proposed, which is a scalable, fine-grained, and lightweight solution for today's IoT networks;
2. A complete architecture of a federated capability-based authorization system is designed, which includes delegation authority, capability management, and access right validation;
3. A capability-based federated delegation model is introduced and the enforcement of polices is discussed in detail;
4. A concept-proof prototype based on smart contracts is implemented on resource-constrained edge devices and more powerful devices, and deployed on a local private blockchain network; and
5. A comprehensive experimental study has been conducted that compares the proposed scheme with the well-known RBAC and ABAC models. The experimental results validate the feasibility of the BlendCAC approach in IoT environments without introducing significant overhead.

The remainder of this paper is organized as follows: Section 2 gives a brief review on the state-of-the-art research in access control for IoT systems. Section 3 defines components of federated delegation model including capability-based delegation and revocation. Then, Section 4 illustrates the details of the proposed BlendCAC system and Section 5 explains the implementation of the proof-of-concept prototype. The experimental results and evaluation are discussed in Section 6. Finally, the summary, current limitations and on-going efforts are discussed in Section 7.

2. Background Knowledge and Related Work

2.1. Access Control in IoTs

Technologies for authentication and authorization of access to certain resources or services are among the main elements to protect the security and privacy for IoT devices [8]. As a fundamental mechanism to enable security in computer systems, AC is the process that decides who is authorized to have what communication rights on which objects with respect to some security models and policies [9]. An effective AC system is designed to satisfy the main security requirements, such as confidentiality, integrity, and availability. However, recently raised security and privacy issues push the AC systems in the era of IoT to meet a higher bar with more design considerations such as high scalability, flexibility, lightweight and causality.

There are various AC methods and solutions with different objectives proposed to address IoT security challenges. The Role-Based Access Control (RBAC) model [10] provides a framework that species user access authorization to resources based on roles, and supports principles such as least privilege, partition of administrative functions and separation of duties [11]. However, a pure RBAC model presents a role explosion problem, which is inappropriate to implement security policies that require interpreting complex and ambiguous IoT scenarios. The RBAC model implemented on devices adopts a Web of Things (WoTs) approach to implement AC policies on the smart objects via the web service [12,13], and the RBAC model was extended by introducing context constraints to consider contextual awareness in AC decisions [14]. However, those proposals are not able to clearly specify the fine-grained AC on variant resources or services, like the mapping of the role notion and device-to-device communication.

To address the weaknesses of RBAC model in a highly distributed network environment, an Attribute-based Access Control (ABAC) [15,16] is introduced in IoT networks to reduce the number of rules resulting from role explosion. In ABAC the AC policies are defined through directly associating attributes with subjects. An efficient authentication and ABAC based authorization scheme for the IoT perception layer has been proposed [17]. Based on user attribute certificates, an access right is granted by AC authority to ensure fine-grained access control. However, specifying a consistent definition of the attributes within a domain or across different domains could significantly increase effort and complexity on policy management as the number of devices grow, and hence, the attribute-based proposal is not suitable for large scale distributed IoT networks.

Due to drawbacks that exist in traditional access control models such as RBAC and ABAC, the requirements imposed by IoT scenarios cannot be satisfied. Given many great advantages from an IoT perspective, such as scalability, flexibility, distributed, and user-driven, IoT systems can support delegation and revocation[8]. Capability-based access control approaches have been considered a promising solution to IoT systems. The Access Control Matrix (ACM) model represents a good conceptualization of authorizations by providing a framework for describing Discretionary Access Control (DAC) [11]. As two implementations of ACM, Access Control List (ACL) and Capability are widely used in authorization systems. In the ACL model, each object is associated with an access control list that saves the subjects and their access rights for the objects.

The ACL is a centralized approach to support administrative activities with better traceability by implementing AC strategies on cloud servers [18]. However, as the number of subjects and resources increases, confused duty problems are identified in ACL and access rules become much more complex to manage. Due to the centralized management property, ACL cannot provide multiple levels of granularity, is not scalable and is vulnerable to a single point of failure. Meanwhile, in the capability-based access control (CapAC) model, each subject is associated with a capability list that represents its access rights to all concerned objects. The CapAC has been implemented in many large scale IoT-based projects, like IoT@Work [19].

Although capability-based methods have been used as a feature in many access control solutions for the IoT-based applications, applying the original concept of capability-based access control model in IoT networks has raised several issues, like capability propagation and revocation [9]. To tackle these challenges, a Secure Identity-Based Capability (SICAP) System is proposed, which enables the monitoring, mediating, and recording of capability propagations to enforce security policies as well as achieving rapid revocation capability by using an exception list [9]. However, the centralized access control server (ACS) becomes the performance bottleneck of the system, and the author didn't provide a clear illustration on security policy used in capability generation and propagation, neither was the context information in making authorization decision considered.

To enable contextual awareness in federated IoT devices, an authorization delegation method is proposed based on a Capability-based Context-Aware Access Control (CCAAC) model [20]. By introducing a delegation mechanism to capability generation and propagation process, the CCAAC model shows great advantages to address scalability and heterogeneity issues in IoT networks. Given the requirement that a prior knowledge of the trust relationship among domains in federated IoTs must be established, however, the proposed approach is not suitable universally for all IoT application scenarios. Inspired by the SUN DIGITAL ECOSYSTEM ENVIRONMENT project [21], a Capability-based Access Control (CapAC) model was proposed that adopted a centralized approach for managing access control policy [5]. However, the proposed CapAC scheme depends on a centralized authority and did not consider the lightweight requirement at the smart device side. To address the limitations in CapAC, a Distributed Capability-based Access Control (DCapAC) model was proposed, which was directly deployed on resource-constrained devices [22,23]. The DCapAC allows smart devices to autonomously make decisions on access rights based on an authorization policy, and it shows advantages in scalability and interoperability. However, capability revocation management and delegation were not discussed, neither were the granularity and context-awareness considered.

2.2. Blockchain and Smart Contract

The blockchain is the fundamental framework of Bitcoin [6], which was introduced by Nakamoto in 2008. The blockchain is the public ledger that allows the data be recorded, stored and updated distributively. By its nature, the blockchain is a decentralized architecture that does not rely on a centralized authority. The transactions are approved and recorded in blocks created by miners, and the blocks are appended to the blockchain in a chronological order. Blockchain uses consensus mechanism to maintain the sanctity of the data recorded on the blocks. Thanks to the "trustless" proof mechanism enforced through mining task on miners across networks, users can trust the system of the public ledger stored worldwide on many different decentralized nodes maintained by ”miner-accountants,” as opposed to having to establish and maintain trust with the transaction counter-party or a third-party intermediary [24]. Blockchain is the ideal architecture to ensure distributed transactions between all participants in a trustless environment.

Because of many attractive characteristics, blockchain technology has been investigated to offer a decentralized AC scheme in trustless network environments. A blockchain based AC is proposed to publish AC policy and to allow distributed transfer access right among users on bitcoin network [25]. The proposal allows distributed auditability, preventing a third party from fraudulently denying the rights granted by an enforceable policy. However, the solution still relies on an external centralized policy database to fetch access right given the links stored in the blockchain, and the experimental results are not provided. Based on blockchain technology, FairAccess is proposed to offer a fully decentralized pseudonymous and privacy preserving authorization management framework for IoT devices [26]. In FairAccess, the AC policies are enclosed in new types of transactions that are used to grant, get, delegate, and revoke access. However, the scripting language used in Bitcoin allows to transcode two types of AC policies, so that the proposed framework cannot support more complex and granular access control model.

Blockchain has shown its success in decentralization of currency and payments, like Bitcoin. Currently, designing a programmable money and contracts, which support variety of customized transaction types, has become a trend to extend blockchain applications beyond the cryptocurrency domain. *Smart contract*, which emerges from the smart property, is a method of using blockchain to achieve agreement among parties, as opposed to relying on third parties for maintaining a trust relationship. By using cryptographic and other security mechanisms, smart contract combines protocols with user interfaces to formalize and secure relationships over computer networks [27]. Smart contract is essentially a collection of pre-defined instructions and data that have been recorded at a specific address of blockchain. Through encapsulating operational logic as a bytecode and performing Turing complete computation on distributed miners, a smart contract allows user to transcode more complex business models as new types of transactions on a blockchain network. Smart contract provides a promising solution to implement more flexible and fine-grained AC models on blockchain networks.

3. Federated Capability-based Delegation and Revocation Model

In today's IoT based systems, data service and security enforcement are deployed on centralized cloud centers where abundant computing and storage resources are allocated. Such a centralized network architecture is not scalable for large-scale IoT networks, and management efforts for resource and security policy become dramatically increased owning to heterogeneity property in a highly decentralized IoT environment. Delegation enables an entity to give permission to let other entities function on its behalf by providing all or some of its rights. It is considered a useful and effective approach to improve the scalability of distributed systems and decentralize access control tasks [28].

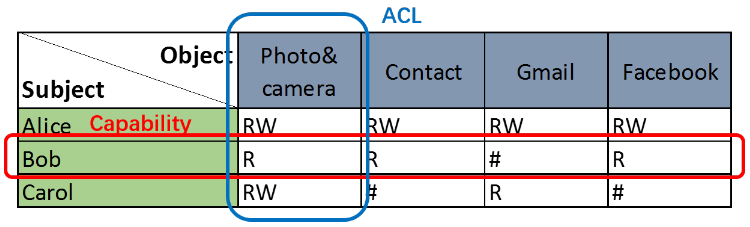
As an important factor for secure distributed system, delegation has been recognized as one of the schemes to support access policy management in a distributed computing environment [29]. Although a rule-based framework for role-based delegation and revocation was proposed [30], the proposal cannot support capability-based access control system. In this section we propose a *Federated Capability-based Delegation Model* (FCDM). This FCDM model supports capability-based hierarchy and multi-step delegation by introducing the delegation relationship.

3.1. Capability Access Control Model

The elements and relations in Capability AC model are depicted in Figure 1. Access Control Matrix (ACM) includes sets of three basic elements: Subject S, Object O, and Permission P. Access Control List (ACL) and Capability are two permission relationships in the ACM model. The ACL permission assignment is a many-to-one relation between Subject and Objects, which means that each object is associated with a set of access control lists that save the subjects and their authorized permissions for the object. However, the capability model uses subject oriented permission assignment in which relations between Subject and Objects becomes one-to-many.

A subject in the capability model is a human being or device, a object is an entity who offers services or resources, and permission refers to an authorized activity to carry out a particular task or an access resource on an object. In Fig. 1, the permissions are: R-read, W-write and #-not allowed. For each subject, the capability specifies a set of connected objects which are associated with authorized permissions to access services or resources. The following is a list of definitions in capability model:

- S, O and P are sets of subjects, objects and permissions.
- ${}_{in}Cap_O \subseteq O \times P$ is internal capability which defines a one-to-many relation assignment between object and permissions.



Subject \ Object	Photo& camera	Contact	Gmail	Facebook
Alice	RW	RW	RW	RW
Bob	R	R	#	R
Carol	RW	#	R	#

ACL
Capability

Figure 1. Capability Access Control Model.

- ${}_{ext}Cap_{(S,O)} \subseteq S \times {}_{in}Cap_O$ is external capability which specifies a one-to-many relation assignment by associating a subject with a subset of internal capabilities.

The ${}_{in}Cap_O$ are defined by object owner or policy authority to indicate all available service permissions supported by an object. The policy decision authority could generate ${}_{ext}Cap_{(S,O)}$ for a subject to access an object by authorizing permissions in ${}_{in}Cap_O$.

3.2. Federated Capability-based Delegation Model

Delegation is an efficient mechanism to simplify access policy management by building a hierarchical relationship to reflect an organization's lines of authority and responsibility. In essential, a delegation hierarchy is a partial order relation $\preceq$. If x delegates the permissions to y, or y inherits the permissions of x, the delegation relation between x and y could be represented as a partial order y$\preceq$x. A partial order is a reflexive, transitive, and antisymmetric relation.

To control the delegation propagation and simplify the federated delegation model, we assume that a subject cannot be delegated any new permissions if the subject has already been assigned delegated permissions. Given the above assumption, *Delegation Relation (DR)* in FCDM is defined as follows:

- $DR \subseteq {}_{ext}Cap \times {}_{ext}Cap$ is one-to-many delegation relation. A delegation relation can be represented by $((S1, (O \times P1)), (S2, (O \times P2))) \in DR$, where P2$\preceq$P1. It indicates that subject S1 delegates subset of P1 to subject S2 as P2.

To confine delegation relation propagation steps, it's necessary to set delegation depth to define maximum times whether or not delegation operation can be further performed. Two types of delegation: *single-step delegation* and *multi-step delegation* are considered in FCDM. Single-step delegation prevents the delegated subject from further performing delegation; whereas multi-step delegation allows multiple delegate operations until it reaches the maximum delegation depth. Thus, Single-step delegation is considered to be a special case of multi-step delegation with maximum delegation depth equal to one.

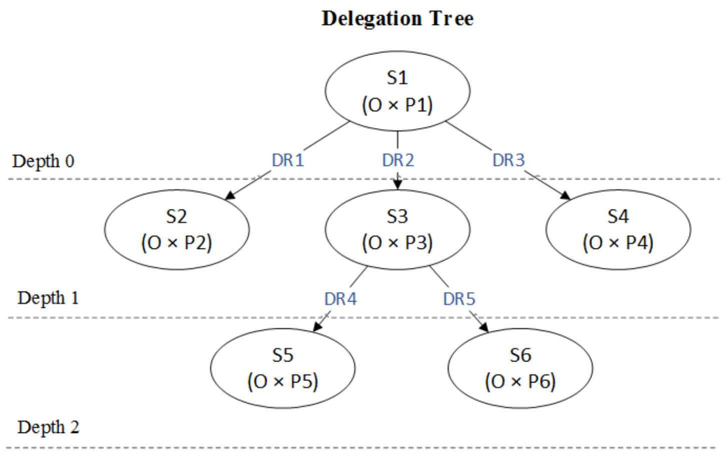
Multi-step delegation generates an ordered list of delegation relation, called *Delegation Path (DP)*. In general, a delegation path starts from an initial or root ${}_{ext}Cap$, and is represented as the following notation:

$$DR_0 \rightarrow DR_1 \rightarrow \cdots \rightarrow DR_i \rightarrow \cdots \rightarrow DR_n$$

All delegation paths starting with the root ${}_{ext}Cap$ construct a hierarchical structure: *Delegation Tree (DT)*. In the delegation tree, each node represents a ${}_{ext}Cap$ and each edge refers to a DR. The layer of ${}_{ext}Cap$ in the tree is defined as the delegation depth.

Given above discussions, the definitions and functions in FCDM are:

- $DP \subseteq DR \times DR$ is an ordered list of delegation relation indicating a delegation path.
- $DT \subseteq DR \times DR$ is a delegation relation hierarchy representing a delegation tree.
- $N_{maxDepth}$ is a natural number representing maximum delegation depth.



Delegation Relation	Delegation Path
DR1	DP1: (S1, (O × P1)) → (S2, (O × P2))
DR2	DP2: (S1, (O × P1)) → (S3, (O × P3))
DR3	DP3: (S1, (O × P1)) → (S4, (O × P4))
DR4	DP4: (S1, (O × P1)) → (S3, (O × P3)) → (S5, (O × P5))
DR5	DP5: (S1, (O × P1)) → (S3, (O × P3)) → (S6, (O × P6))

Figure 2. An Example for Delegation Paths and an Delegation Tree.

- *Ancestor*: ${}_{ext}Cap \rightarrow {}_{ext}Cap$ is a function that maps a ${}_{ext}Cap$ node to another parent ${}_{ext}Cap$ node in the delegation tree.
- *Path*: ${}_{ext}Cap \rightarrow DP$ is a function that maps a ${}_{ext}Cap$ node to a delegation path. Path(${}_{ext}Cap_i$) = { $DR_0 \rightarrow DR_1 \rightarrow \cdots \rightarrow DR_i \mid {}_{ext}Cap_i = Ancestor({}_{ext}Cap_{i-1})$}
- *DelegateDepth*: ${}_{ext}Cap \rightarrow N$ is a function that returns a delegation depth in delegation tree given ${}_{ext}Cap$ node, where N is a natural number set representing delegation depth.

In order to illustrate the concepts of delegation path and delegation tree. a set of delegation relations example is list as follows:

$$
\begin{aligned}
DR1 &: ((S1,(O \times P1)),(S2,(O \times P2))) \in DR \\
DR2 &: ((S1,(O \times P1)),(S3,(O \times P3))) \in DR \\
DR3 &: ((S1,(O \times P1)),(S4,(O \times P4))) \in DR \\
DR4 &: ((S3,(O \times P3)),(S5,(O \times P5))) \in DR \\
DR5 &: ((S3,(O \times P3)),(S6,(O \times P6))) \in DR
\end{aligned}
$$

According to above delegation relations, all delegation paths can be calculated by applying *Path* function, and build up a delegation tree as shown in Fig 2. In the delegation tree, the parent node only delegated a subset of its permissions to a child, so that permission propagation over delegation path is essentially a partial order sequence. Take DP5 for example, permission delegation is represented as partial order relation: $P6 \preceq P3 \preceq P1$.

3.3. Capability-based Delegation Authorization

Delegation authorization is mainly to impost restrictions on which subject can be delegated to whom based on delegation authorization rules. In our proposed FCDM, the subject-to-subject delegation authorization relation is defined as follows:

- ${}_{ext}Cap$, S, C, $N_{maxDepth}$ are sets of capability, subject, conditions for authorization and maximum delegation depth, respectively.
- $can\_Delegate \subseteq {}_{ext}Cap \times S \times C \times N_{maxDepth}$.

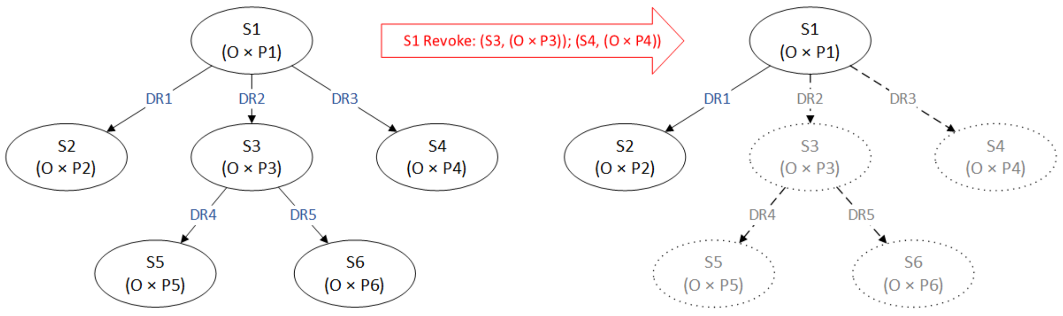


Figure 3. An Example for Grant-independent Cascading Revocation.

The relation $({}_{ext}Cap_{S1},\ S2,\ C,\ N_{maxDepth}) \in can\_Delegate$ means that a subject $S1$, who is a node in delegation tree with capability ${}_{ext}Cap_{S1}$, could delegate its subset of permissions to subject $S2$ whose properties satisfy the conditions without exceeding the maximum delegation depth $N_{maxDepth}$. Take an example in Fig. 2, $({}_{ext}Cap_{S4} \in {}_{ext}Cap\ ) \wedge (({}_{ext}Cap_{S4},\ S7,\ C,\ 1) \in can\_Delegate)$, where $N_{maxDepth} = 3$ and $C = \{{}_{ext}Cap_{S7} \notin {}_{ext}Cap\}$, then $S4$ can delegate subset of its permission $P4$ to new subject $S7$ as permission $P7$. As a result, the new delegation relation $DR6 : ((S4,(O \times P4)),(S7,(O \times P7))) \in DR$ and capability ${}_{ext}Cap_{S7} = (S7,(O \times P7)) \in {}_{ext}Cap$ are created and appended to DR3 as a leaf node of delegation tree DT.

3.4. Capability-based Delegation Revocation

As an important process that accompanies the delegation mechanism, revocation refers to process to nullify the delegated permissions, or attempts to rollback the state before permissions were delegated. The revocation approaches can be categorized into three dimensions [30]: *grant-dependency*, *propagation*, and *dominance*. Our FCDM only considers two dimensions: *grant-dependency* and *propagation*.

Grant-dependency refers to the legitimacy of a subject who can take away assigned permissions from a delegated subject, and has two types: *grant-dependent* and *grant-independent*. *Grant-dependent* revocation means that only the delegating subject (parent) can revoke the permissions from directly delegated subjects (children). *Grant-independent* revocation means any ancestor subject in the delegation path can revoke the delegated permissions from the offspring subjects.

Propagation specifies the extent of the revocation to subsequent delegated subjects. it can be categorized as *cascading* and *non-cascading*. *Cascading* revocation directly revokes delegated permissions from subject as well as indirectly nullified a set of subsequent propagated delegation relation. While *Non-cascading* revocation only takes away directly delegated permissions from children subjects.

To reduce the complexity in the revocation process, our FCDM enforces *grant-independent* and *cascading* rules in delegation revocation. Figure 3 is an example for grant-independent cascading revocation. Revocation authorization can be defined as follows:

- ${}_{ext}Cap,\ S,\ Ancestor()$ are sets of capability, subject and Ancestor function, respectively.
- $can\_Revoke \subseteq {}_{ext}Cap \times S \times Ancestor(S)$.

The relation $({}_{ext}Cap_{S1},\ Ancestor(S2)) \in can\_Revoke$ means that a subject S1, who is the ancestor of subject S2, can revoke delegated permissions of S2 as well as all indirect assigned permissions by S2 in subsequent delegation relation. As shown in Fig. 3, owing to fact that both $S1 \in Ancestor(S3)$ and $S1 \in Ancestor(S4)$, revocation authorization satisfies the relation $({}_{ext}Cap_{S1},\ Ancestor(S3)) \vee ({}_{ext}Cap_{S1},\ Ancestor(S4)) \in can\_Revoke$. As a result, the delegation relation DR2 and DR3 are removed from delegate tree and delegated capability ${}_{ext}Cap_{S3}$ and ${}_{ext}Cap_{S4}$ are revoked. In addition, subsequent relation DR4 and DR5 assigned by S3 are also removed, and associated capability ${}_{ext}Cap_{S5}$ and ${}_{ext}Cap_{S6}$ are revoked.

4. BlendCAC: a BLockchain-ENabled Decentralized Federated CapAC System

Most dominant IoT based systems, like ASW IoT, utilize a centralized cloud platform where abundant computing and storage resources are allocated to securely manage connected devices. As a result, all service access requests on devices need to be transmitted to remote servers for authentication and authorization. Such a centralized network architecture is not scalable for today's large IoT networks, and latencies are not tolerable in many mission-critical applications. To meet the requirements of real-time processing and instant decision making, online, uninterrupted smart surveillance systems have been intensively studied recently leveraging the edge-fog-cloud computing paradigm [31–33]. Based on an automatic surveillance system architecture, a federated capability-based access control system (FedCAC) [34] has been proposed to addresses scalability, granularity, and dynamicity challenges in access control strategy for IoT devices. Through delegating part of the identify authentication and authorization task to domain delegator, workload of the centralized policy decision making center (PDC) is reduced. Migrating some processing validation tasks to local devices helps the FedCAC to be lighter and context-awareness enabled. Involving smart objects in access right authorization process allows device-to-device communication, which implies better scalability and interoperability in an IoT network environment. However, a comprehensive capability-based delegation and revocation mechanism is not illustrated. In addition, FedCAC is essentially still a centralized AC scheme, such that weaknesses include being the single-point of failure and performance bottleneck, are still not solved.

Inspired by the smart contract and blockchain technology, a decentralized federated capability-based access control framework for IoTs, called BlendCAC, is proposed in this paper, and a prototype of proposal has been implemented in a physical IoT network environment to verify the efficiency and effectiveness. The next subsection provides a comprehensive system design of BlendCAC framework. Unlike the approaches discussed above, BlendCAC, effectively provides decentralization, scalability, granularity, and dynamicity of AC strategies for IoTs. Through encapsulating FCDM model and AC policies into a Smart Contract, which is deployed across the blockchain network, users are the master of their own data or devices instead of being supervised or controlled by a third party authority. Enforcing authorization and access right verification among a large number of distributed edge devices allows more coordination on edge networks.

4.1. System Architecture of BlendCAC

Figure 4 illustrates the proposed BlendCAC system architecture, which intends to function in a scenario including two isolated IoT-based service domains without pre-establishing a trust relationship. In our proposed BlendCAC framework, the cloud works as service provider to provide global profile data and security policy management , and the domain coordinator enforces delegated security policies to manage domain related devices and services. Operation and communication modes are listed as follows:

1. *Registration*: All entities must create at least one main account defined by a pair of keys to join the blockchain network. Each account is indexed by its address that is derived from his/her own public key. In our scenario, identity authentication and management is deployed on two levels: cloud and coordinator. The cloud server maintains a global profile database, and the domain coordinator maintains a local profile database, and regular synchronization between the cloud server and domain coordinator ensures data consistence. New users could either sends registration request to cloud or delegated coordinator. Once the identity information related to users or IoT devices is verified, the profile of each registered entity is created by using his/her address for authentication process when an access right request happens. As a result, the domain coordinators are able to enforce delegated authorization policies and perform decision-making to directly control their own devices or resources instead of depending on third parties.

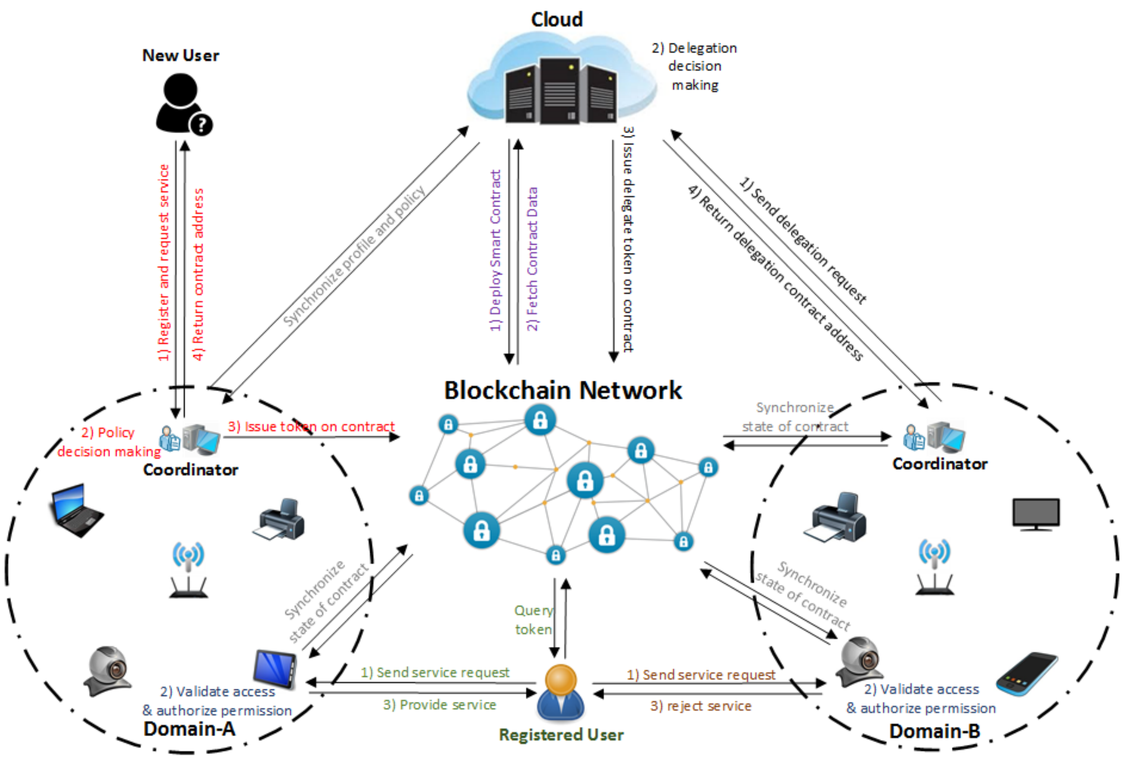

Cloud
2) Delegation decision making
New User
1) Register and request service
4) Return contract address
Synchronize profile and policy
1) Deploy Smart Contract
2) Fetch Contract Data
3) Issue delegate token on contract
1) Send delegation request
4) Return delegation contract address
Blockchain Network
2) Policy decision making
Coordinator
3) Issue token on contract
Synchronize state of contract
Coordinator
Synchronize state of contract
Synchronize state of contract
Query token
1) Send service request
3) Provide service
1) Send service request
3) reject service
2) Validate access & authorize permission
Domain-A
Registered User
2) Validate access & authorize permission
Domain-B

Figure 4. System Architecture of BlendCAC.

2. *Smart Contract Deployment*: A smart contract, which manages federated delegation relation and capability tokens, must be developed and deployed on the blockchain network by the policy owner. In our framework, the cloud acts as data and policy owner who could deploy smart contract encapsulating delegation and CapAC token. Thanks to cryptographic and security mechanisms provided by blockchain network, smart contracts can secure any algorithmically specifiable protocols and relationships from malicious interference by third parties under trustless network environment. After synchronizing the blochchain data, all nodes could access all transactions and recent state of each smart contract by referring local chain data. Each node interacts with the smart contract through the provided contract address and the Remote Procedure Call (RPC) interface.
3. *Federated Delegation*: The PDC at the cloud server is responsible for delegation policy definition and access right authorization enforcement. To reduce the overhead of the centralized cloud server and meet requirements of scalability and heterogeneity in each IoT domain, the domain coordinator delegates part of the policy decision making tasks and carries out domain specified authorization rules based on domain specified policies. After receiving a delegation request from a coordinator candidate and executing policy decision making task to delegate permissions to coordinator, cloud launches a transaction to issue a delegation certificate on smart contract. Finally, the federated delegation relationship is established between the cloud server and the coordinator, and profile and policy data synchronization between the cloud and coordinator is periodically carried out to ensure data consistence on two side.
4. *Capability Authorization*: To successfully access services or resources at service providers, an entity initially sends an access right request to the domain coordinator to get a capability token. Given the registered entity information established in the profile database, a policy decision making module evaluates the access request by enforcing the delegated authorization policies. If the access request is granted, the domain coordinator issues the capability token encoding the access right, and then launches a transaction to update the token data in the smart contract. After the

transaction has been approved and recorded in a new block, the domain coordinator notifies the entity with a smart contract address for the querying token data. Otherwise, the access right request is rejected.

5. *Access Right Validation*: The authorization validation process is performed at the local service providers on receiving a service request from the entity. Through regularly synchronizing the local chain data with the blockchain network, a service provider just simply checks the current state of the contract in the local chain to get a capability token associated with the entity's address. Considering the capability token validation and access authorization process result, if the access right policies and conditional constraints are satisfied, the service provider grants the access request and offers services to the requester. Otherwise, the service request is denied.

To enable a scalable, distributed and fine-grained access control solution to IoT networks, the proposed BlendCAC is focused on three issues: identity-based capability management, access right authorization and privilege mechanism delegation.

4.2. Capability Token Structure

In the BlendCAC system, the entities are categorized as subjects and objects. *Subjects* are defined as entities who request a service from the service providers, while *objects* are referred to entities who offer the resources or services. Entities could be either human beings or smart devices. In the profile database, all registered entities are associated with a globally unique Virtual Identity (VID), which is used as the prime key for identifying entities' profile information. As each entity has at least one main account indexed by its address in the blockchain network, the blockchain is used to represent the VID for profiling register entities.

In general, the capability specifies which subject can access resources of a target object by associating subject, object, actions and condition constraints. The identity-based capability structure is defined as follows:

$$ICap = f(VID_S, VID_O, AR, C) \tag{1}$$

where the parameters are:

- f: a one-way hash mapping function;
- VID_S: the virtual ID of a subject that requests an access to a service or resource;
- VID_O: the virtual ID of an object that provides a service or resource;
- AR: a set of access right for actions, e.g. read, write, execute; and
- C: a set of context awareness information, such as time, location.

In the BlendCAC system, an AR is defined as the access right set. For example, the AR can be $\{Read\}$, $\{Write\}$, $\{Read; Write\}$, or $\{NULL\}$. If $AR = \{NULL\}$, the operation conducted on the resource is not allowed. C is defined as a context constraints set, like $C = \{C1, C2\}$ or $C = \{NULL\}$. If $C = \{NULL\}$, no context constraint is considered in the access right validation process.

4.3. Delegation Certificate Structure

Identity-based Delegation Certificate (IDC) is in essential a special capability token which specifies the delegation relation. The structure of IDC is represented as follows:

$$IDC = f(VID_S, \{VID_P\}, \{VID_C\}, D, W, DAR) \tag{2}$$

where the parameters are:

- f: a one-way hash mapping function;
- VID_S: the virtual ID of a subject who is the owner of the delegation token;
- VID_P: the virtual ID of a parent subject that delegates the token to VID_S;
- $\{VID_C\}$: a set of virtual ID of children subject that records the delegated nodes;

- D: a natural number that indicates the current depth in delegation tree;
- W: a natural number that defines maximum delegation width to limit delegable children nodes in $\{VID_C\}$; and
- DAR: a set of delegated permissions for actions, e.g. authorize capability token.

In order to manage delegation relations between IDC, a hierarchical data structure, called *Identity-based Delegation Tree* (IDT) is defined as follows:

$$IDT = f(S, MD, IDC) \tag{3}$$

where the parameters are:

- f: a one-way hash mapping function;
- S: the virtual ID of a subject who is the owner of delegation tree;
- MD: a natural number that defines maximum delegation depth; and
- IDC: a delegation certificate that indicates the root node of delegation tree.

4.4. Federated Delegation Mechanism

Through encapsulating a delegation certificate structure as smart contract and deployed on blockchain network, the delegation mechanism could be enforced across different security domains in a federated network environment. In the BlendCAC system, the delegator, delegation authority center (DAC) and identity management are all implemented as service applications on the cloud server, while the delegatee is deployed on the coordinator in each network domain. The DAC is responsible for identification authentication and delegation authorization service. Prior to the delegation process, the delegator and delegatee should have finished the identity registration process. Figure 5 illustrates the delegation process in our proposed BlendCAC system. The involved work flow in delegation process is:

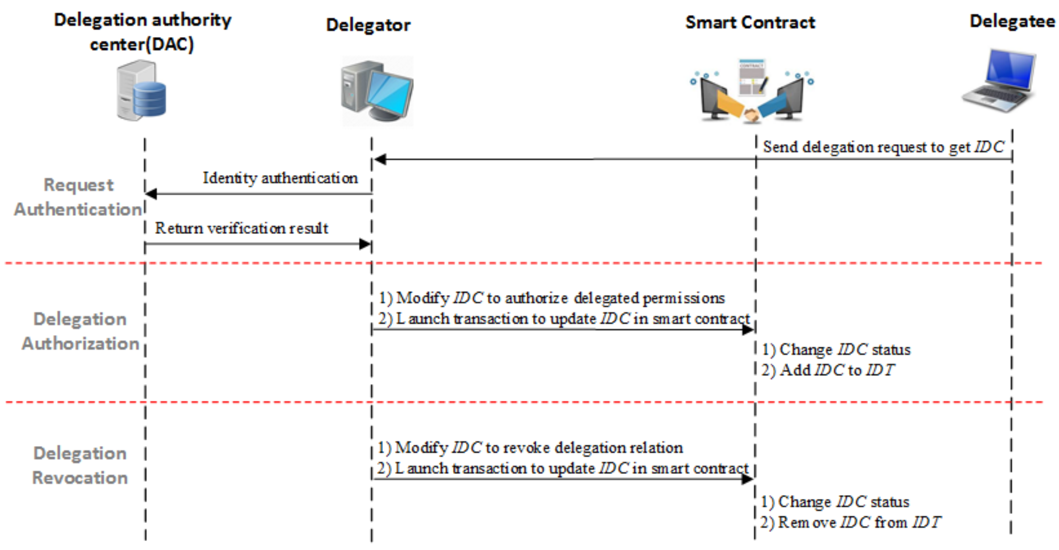

Delegation authority center(DAC)
Delegator
Smart Contract
Delegatee
Send delegation request to get IDC
Request Authentication
Identity authentication
Return verification result
Delegation Authorization
1) Modify IDC to authorize delegated permissions
2) Launch transaction to update IDC in smart contract
1) Change IDC status
2) Add IDC to IDT
Delegation Revocation
1) Modify IDC to revoke delegation relation
2) Launch transaction to update IDC in smart contract
1) Change IDC status
2) Remove IDC from IDT

Figure 5. Delegation Process in BlendCAC System.

- *Request Authentication*: The delegatee sends a delegation request to the delegator to ask for IDC. On receiving the request from the delegator, the DAC verifies the identity of delegatee by referring identity management service. If the delegatee's identity is valid, the identity management service returns a virtual ID (VID) of the delegatee to the DAC. Then the DAC sends back the authentication result to the delegator. Otherwise, the DAC rejects the delegation request by returning a failure notification.

- *Delegation Authorization*: After receiving the verification result from the DAC, the delegator is capable of assigning delegable permissions to trusted delegatee given predefined delegation policies. In our proposal, the federated delegation mechanism is implemented by modifying delegation certificate IDC to assign the delegated permissions or change delegation relation. As the smart contract has received an update DC transaction from the delegator, it checks the delegator's IDC to validate the delegation right. If the delegation status can match the $can\_Delegate$ relation, the delegator could delegate subset of his/her delegated permissions to delegatee entity by modifying delegatee's IDC and appending the delegatee's address to delegator's children nodes set $\{VID_C\}$ to set up delegation relation. Otherwise, the capability delegation request is rejected.
- *Delegation Revocation*: The delegation revocation considers two scenarios: delegated permissions P revocation and delegation certificate IDC revocation. According to revocation mechanism defined in FCDM, only entities who meet the can_Revoke relation could carry out revocation operation over smart contract. In delegated permissions revocation process, the delegator could nullify part of assigned permissions by simply removing access right elements from DAR in delegatee's IDC. In case of IDC revocation, through *cascading* removing all subsequent delegate relations from DP which starts from delegator and destructing delegatee's IDC, the delegator could tear down all delegation relations that is associated with delegatee, including those $IDCs$ assigned by delegatee.

4.5. Capability-based Access Right Authorization

The capability token structure and the related operations are transcoded to a smart contract and deployed on the blockchain network, while the access right authorization is implemented as a policy-based decision making service running on the cloud or delegated domain coordinator. As shown by Fig 6, a comprehensive capability-based access right authorization procedure consists of four steps: capability generation, access right validation, capability delegation and revocation.

1. *Capability Generation*: As one type of meta data to represent the access right, the capability $ICap$ could be generated by associating a VID with an AR, thus the $ICap$ has the identified property to prevent forgery. After receiving access request from user, the domain coordinator generates capability token based on delegated access right authorization policy, and launches transactions to save a new token data to a smart contract. A large number of $ICap$'s are grouped into the capability pools on smart contract, which could be proofed and synchronized among the nodes across the blockchain network.
2. *Access Right Validation*: After receiving the service request from a subject, the service provider first fetches the capability token from the smart contract by using the subject's address, then makes decisions on whether or not to grant an access to the service according to the local access control policy. Implementing access right validation at the local service provider allows smart objects to be involved in the AC decision making task, which is suitable to offer a flexible and fine-grained AC service in IoT networks.
3. *Capability Revocation*: The capability revocation considers two scenarios: partial access right revocation and $ICap$ revocation. In our proposal, only the entities with delegated capability management permissions are allowed to perform revocation operation on capability tokenized smart contract. In the partial access right revocation process, the delegated entities could remove part of entries from AR to revoke the selected access right. In case of $ICap$ revocation, through directly clearing the AR in $ICap$, the whole capability token becomes unavailable to all associated entities.

5. Prototype Design

A concept-proof prototype system has been implemented on a real private Ethereum blockchain network environment. Compared with other open blockchain platforms, like Bitcoin and Hyperledger,

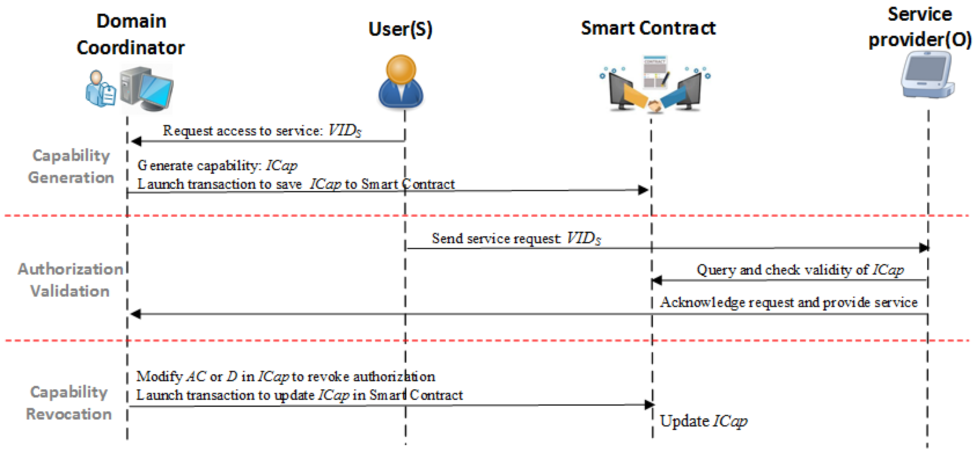

Domain Coordinator
User(S)
Smart Contract
Service provider(O)
Capability Generation
Request access to service: VID_S
Generate capability: $ICap$
Launch transaction to save $ICap$ to Smart Contract
Authorization Validation
Send service request: VID_S
Query and check validity of $ICap$
Acknowledge request and provide service
Capability Revocation
Modify AC or D in $ICap$ to revoke authorization
Launch transaction to update $ICap$ in Smart Contract
Update $ICap$

Figure 6. Flowchart of the Capability-based Access Right Authorization.

Ethereum has a more matured ecosystem and is designed to be more adaptable and flexible for the development of a smart contract and business logic [35].

5.1. Delegation Certificate and Capability Token Structure

The proposed BlendCAC model has been transcoded to a smart contract using Solidity [36], which is a contract-oriented, high-level language for implementing smart contracts. With Truffle [37], which is a world class development environment, testing framework and asset pipeline for Ethereum, contract source codes are compiled to Ethereum Virtual Machine (EVM) bytecode and migrated to the Ethereum blockchain network.

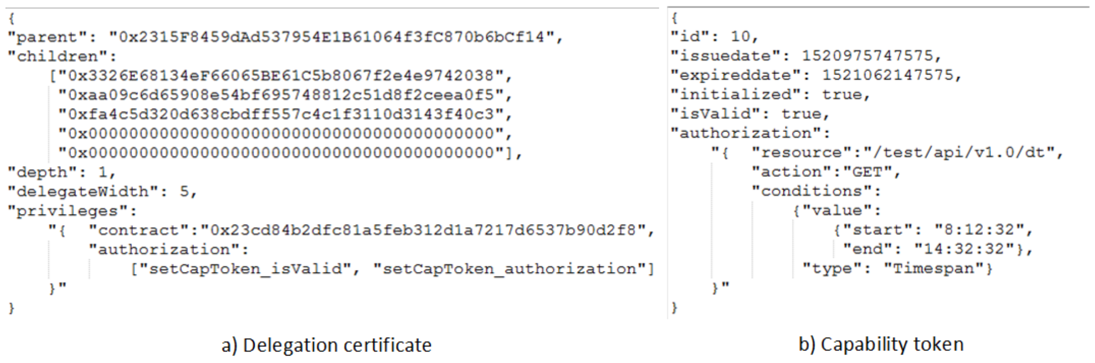
To implement a BlendCAC system on IoT devices without introducing significant overhead over network communication and computation, delegation certificate and capability token data structure is represented in JSON [38] format. Compared to XML-based language for access control, like XACML and SAML, JSON is lightweight and suitable for constrained platforms.

Figure 7: a) demonstrates a delegation certificate example, and the data fields in the data structure are described as follow:

- *parent* : a 20 bytes value to represent address of parent node in blockchain network;
- *children*: a queue to record all address of delegated entities;
- *depth*: a natural number to indicate depth of current delegation certificate in the delegation tree;
- *delegateWidth*: a natural number to constraint horizontal delegation times;
- *privileges*: a set of delegated access rights that delegator has assigned to the delegatee, including
  - *contract*: a 20 bytes value to indicate address of delegated smart contract; and
  - *authorization*: a set of delegated functions for which the operations are granted.

Figure 7: b) presents a capability token data example used in the AC system. A brief description of each field is provided as follows:

- *id*: the auto-incremented prime key to identify a capability token;
- *initialized*: a bool flag used for checking token initialized status;
- *isValid*: a bool flag signifying enabled status to show whether token is valid or not;
- *issuedate*: for identifying the date time when the token was issued;
- *expireddate*: the date time when token becomes expired;
- *authorization*: a set of access right rules that the issuer has granted to the subject, including
  - *action*: to identify a specific granted operation over resource;
  - *resource*: the resource in the service provider for which the operation; is granted. In this case, resource is defined as granted REST-ful API; and



```
{
"parent": "0x2315F8459dAd537954E1B61064f3fC870b6bCf14",
"children":
    ["0x3326E68134eF66065BE61C5b8067f2e4e9742038",
     "0xaa09c6d65908e54bf695748812c51d8f2ceea0f5",
     "0xfa4c5d320d638cbdff557c4c1f3110d3143f40c3",
     "0x0000000000000000000000000000000000000000",
     "0x0000000000000000000000000000000000000000"],
"depth": 1,
"delegateWidth": 5,
"privileges":
    "{  "contract":"0x23cd84b2dfc81a5feb312d1a7217d6537b90d2f8",
        "authorization":
            ["setCapToken_isValid", "setCapToken_authorization"]
    }"
}
```

a) Delegation certificate

```
{
"id": 10,
"issuedate": 1520975747575,
"expireddate": 1521062147575,
"initialized": true,
"isValid": true,
"authorization":
    "{  "resource":"/test/api/v1.0/dt",
        "action":"GET",
        "conditions":
            {"value":
                {"start": "8:12:32",
                 "end": "14:32:32"},
            "type": "Timespan"}
    }"
}
```

b) Capability token

Figure 7. Token Data Structure in BlendCAC.

  - *conditions*: a set of conditions which must be fulfilled locally on the service provider to grant the corresponding operation.

After a smart contract has been successfully deployed on the blockchain network, all nodes in the network could interact with smart contract using address of contract and Application Binary Interface (ABI) definition, which describes the available functions of a contract.

5.2. Access Authorization Service

The access authorization and validation policy is enforced as a web service application based on the Flask framework [39] using Python. The Flask is a micro-framework for Python based on Werkzeug, Jinja 2 and good intentions. The lightweight and extensible micro architectures make the Flask a preferable web solution on resource constrained IoT devices.

Web service application in BlendCAC system consists of two parts: client and server. The client performs operation on resource by sending data request to the server, while the server provides REST-ful API for the client to obtain data or perform operation on resource in server side. A capability based access control scheme is enforced on server side by performing access right validation on the service provider. The access right validation process is launched after a request containing the capability token is received on server. Figure 8 shows a block diagram with the steps to process an authorization decision.

1. *Check cached token data*: After receiving a service request from a user, the service provider firstly checks whether or not the token data associated with user's address exists in the local database. If it is failed in searching the token data, the service provider can fetch the token data from the smart contract through calling an exposed contract method and save token data to the local database. Otherwise, the token data is directly reloaded from the local token database for further validation process. The service provider regularly synchronizes the local database with smart contract to ensure the token data consistence.
2. *Verify token status*: As a capability token has been converted to JSON data, the first step of token validation is checking the current capability token status, such as initialized, isValid, issuedate, and expireddate. If any status of a token is not valid, the authorization process stops and sends deny access request acknowledgement back to the subject.
3. *Check whether access is granted or not*: The service provider will go through all access rules in the access right set to guarantee that the request operation is permitted. The process checks whether or not the REST-ful method used by the requester matches the authorized action of current access rules and the value of resource field is the same as the Request-URI option used by the requester. If current access rule verification failed, process skips to the next access rule for evaluation. If

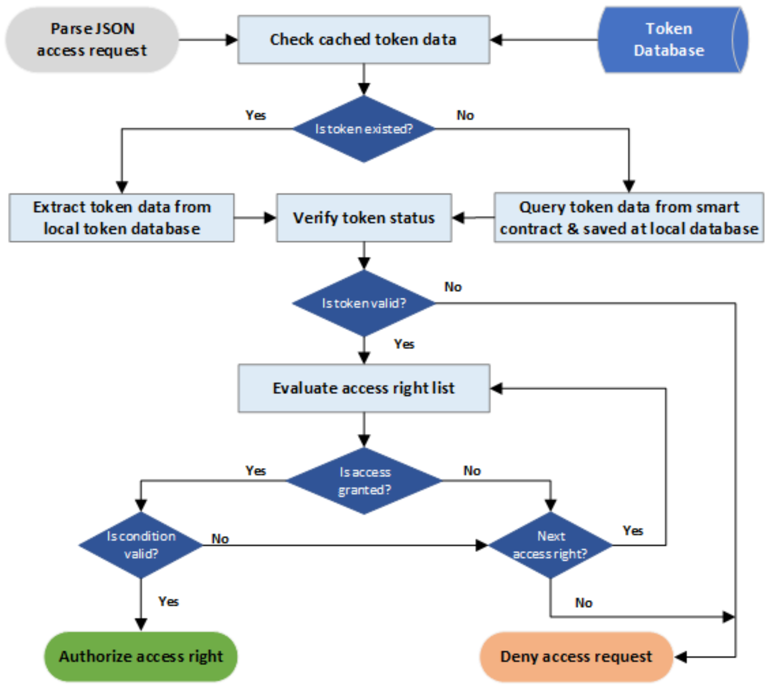


Figure 8. BlendCAC Access Authorization Process.

none of the access rules could successfully pass the verification, the authorization validation process stops and denies the access request.

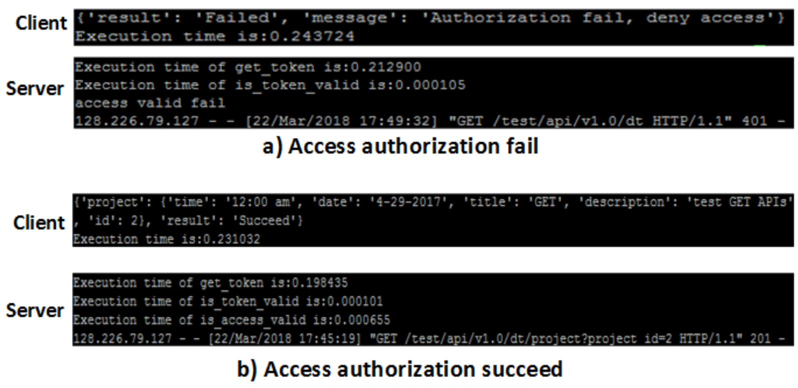
4. *Verify the conditions*: Even through the action on a target resource is permitted after the access validation, the context-awareness constrains are necessary to be evaluated on the local device by verifying whether or not the specified conditions in the token is satisfied. The condition verification process goes through all constraints in the condition set to find the matched ones. If no condition is fulfilled in the given local environment, the access right validation process stops and denies access request.

6. Experimental Results

In order to evaluate the performance and the overhead of our BlendCAC scheme, two benchmark models, RBAC and ABAC, are also transcoded to separate smart contracts and enforced on the experimental web service system. All transcoded access control models have the similar data structure in smart contract except authorization representation. In RBAC based smart contract, authorization is defined as the approach to bridge the relationship between user and permission, while RBAC based smart contract uses user's attributes as representative format for authorization. Both the RBAC and ABAC need a local database, either to maintain the user-role-permission or to manage the attribute-permission policy for authorization validation process, the profiles and policy rules management are developed by using an embedded SQL database engine, called SQLite[40]. The lower memory and computation cost make the SQLite an ideal database solution to resource constrained system, like Raspberry Pi.

6.1. Environmental Setup

The mining task is performed on a system with stronger computing power, like a laptop or a desktop. Two miners are deployed on a laptop, of which the configuration is as follows: the processor is 2.3 GHz Intel Core i7 (8 cores), the RAM memory is 16 GB and the operating system is Ubuntu 16.04. And other four miners are distributed to four desktops which are empowered with the Ubuntu



```
{'result': 'Failed', 'message': 'Authorization fail, deny access'}
Execution time is:0.243724
```

```
Execution time of get_token is:0.212900
Execution time of is_token_valid is:0.000105
access valid fail
128.226.79.127 - - [22/Mar/2018 17:49:32] "GET /test/api/v1.0/dt HTTP/1.1" 401 -
```

```
{'project': {'time': '12:00 am', 'date': '4-29-2017', 'title': 'GET', 'description': 'test GET APIs'
, 'id': 2}, 'result': 'Succeed'}
Execution time is:0.231032
```

```
Execution time of get_token is:0.198435
Execution time of is_token_valid is:0.000101
Execution time of is_access_valid is:0.000655
128.226.79.127 - - [22/Mar/2018 17:45:19] "GET /test/api/v1.0/dt/project?project id=2 HTTP/1.1" 201 -
```

Figure 9. Experimental Results of BlendCAC System.

16.04 OS, 3 GHz Intel Core TM (2 cores) processor and 4 GB memory. In our system, the laptop acts as a cloud computing server, while all desktops work as fog computing nodes to take role of domain coordinator. Each miner uses two CPU cores for mining. The edge computing nodes are two Raspberry PI 3 Model B with the configuration as follows: 1.2GHz 64-bit quad-core ARMv8 CPU, the memory is 1GB LPDDR2-900 SDRAM and the operation system is Raspbian based on the Linux kernel. Unfortunately, the Raspberry PI is not powerful enough to function as a miner, so all Raspberry Pi devices worked as nodes to join the private blockchain without mining. All devices use Go-Ethereum [41] as the client application to work on the blockchain network.

6.2. Experimental Results

To verify effectiveness of our proposed BlendCAC approaches to defense unauthorized access request, a service access experiment is carried out on the real network environment. In the test scenario, one Raspberry Pi 3 device works as the client while another Raspberry Pi 3 device works as the service provider. Given the authorization process shown in Fig 9, when any of the steps in the authorization is failed, the running process will immediately be aborted instead of continuing to carry out all authorization stages. As shown by Fig 9 (a), the server aborted authorization process due to failing to verify granted actions or conditional constraints that are specified in the access right list. As a result, the client node received a deny access notification from the server and cannot read the requested data. In a contrast, Fig 9 (b) presents a successful data request example, in which the whole authorization process is accomplished at the server side without any error. Finally, the client successfully retrieves the data from the service provider.

6.3. Performance Evaluation

In the test scenario, two Raspberry Pi 3 devices are adopted to play the roles of the client and the service provider respectively. To measure the general cost incurred by the proposed BlendCAC scheme both on the IoT devices' processing time and the network communication delay, 50 test runs have been conducted based on the proposed test scenario, where the client sends a data query request to the server for an access permission. This test scenario is based on an assumption that the subject has a valid capability token when it performs the action. Therefore, all steps of authorization validation must be processed on the server side so that the maximum latency value is computed.

6.3.1. Computational Overhead

According to the results shown in Fig. 10, the average total delay time required by the BlendCAC operation of retrieving data from the client to server is 243 ms, which is almost the same as RBAC or ABAC does. The total delay includes the round trip time (RTT), time for querying capability data from

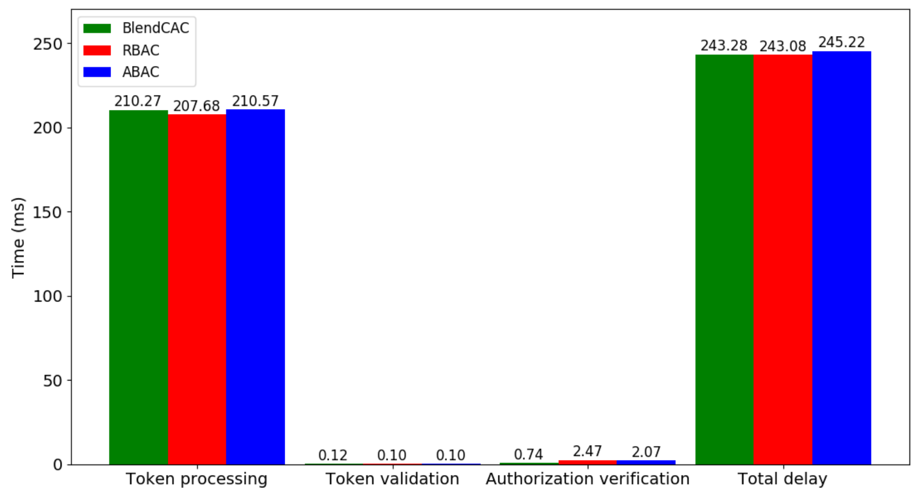


Figure 10. Computation Time for Each Stage in BlendCAC.

the smart contract, time for parsing JSON data from the request, and time for access right validation. The token processing task is mainly responsible for fetching token data from the smart contract and introduces the highest workload among the authorization operation stages. Owing to the fact that encapsulating user-role relationship in the smart contract requires less data than the capability or attributes does, the RBAC incurred less computational cost than what BlendCAC and ABAC did in token processing stage. As the most computing intensive stage, the execution time of token processing is about 210 ms, which is accounted for almost 86% of the entire process time.

The entire authorization process is divided into two steps, token validation and authorization verification, where the average time of the authorization process is about 0.86 ms (0.12 ms + 0.74 ms). Compared with token validation process, which only simply checks the token valid status, the authorization verification process requires more computational power to enforce the local access control policies. Although the similarity in token data structure allows all the three access control models have almost the same time in the token validation stage, the BlendCAC outperforms the RBAC and ABAC in authorization verification. Since both the RBAC and ABAC needs a database to either manage user-role-permission relationship or maintain attributes-permission rules, which inevitably incurs time consuming on searching rules in database. In our experimental study, the RBAC (2.47 ms) and ABAC (2.07 ms) have much higher processing time than BlendCAC (0.74 ms).

6.3.2. Communication Overhead

Owing to the high overhead introduced by querying token data from the smart contract in token processing stage, a token data caching solution is introduced in the BlendCAC system to reduce network latency. When the client sends a service request to the server, the service side extracts cached token data from the local storage to valid authorization. The service providers regularly update cached token data by checking smart contract status. The token synchronization time is in consistence with block generation time, which is about 15 seconds in the Ethereum blockchain network. Simulating a regular service request allows us to measure how long it takes for the client to send a request and retrieve the data from the server.

Figure 11 shows the overall network latency incurred and compares the execution time of the BlendCAC with RBAC, ABAC and a benchmark without any access control enforcement. At the beginning, a long delay is observed in the first service request scenario, in which service provider communicated with the smart contract and cached the token data. However, through processing the local cached token data for authorization validation, the network latency decreases quickly and

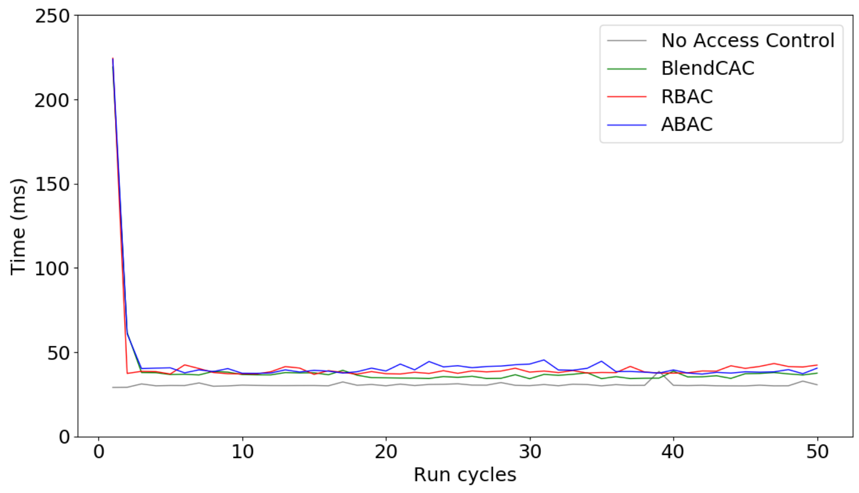


Figure 11. Network Latency of BlendCAC.

becomes stable at a low level during the subsequent service requests. The benchmark without access control enforcement takes an average of 31 ms for fetching requested data versus that the BlendCAC consumes on average of 36 ms. It means that the proposed BlendCAC scheme only introduces about 5 ms extra latency. The overhead in terms of delay is trivial. As shown by Fig. 11, the BlendCAC also has lower latency than RBAC and ABAC in most period of time. In addition, unlike RBAC and ABAC, which rely on the local policy database as intermediate to valid access right, encoding access right directly in capability token makes the BlendCAC more scalable and flexible in the large scale IoT networks.

As the experimental results show, the proposed BlendCAC scheme introduced a small amount of overhead, both at the network layer and the local device layer. To measure general network latency of inter-domain communication, HTTP is executed on the same testbed to simulate a regular transaction, like connects, sends a request and retrieves the reply. Compared with calculated average network latency that is about 300 ms, the trade-off in the proposed BlendCAC is acceptable for the network environments by only incurring 5 ms latency (no more than 2%). In addition, the test scenarios are based on Raspberry Pi devices, which belongs to a type of simple board computer (SBC) with limited computation power and memory space. It is reasonable to expect a better performance when the BlendCAC scheme is implemented on more powerful smart devices, like a smart phone. Although synchronizing cached token data with the smart contract requires more computational resources, the transactions of querying smart contract status are regularly launched by the service providers in a separate service thread rather than being called in each service request, so that network overhead over service request communication is greatly reduced to improve QoS requirement.

6.4. Discussions

The experimental results demonstrate that our proposed BlendCAC strategy is effective and efficient to protecting the IoT devices from an unauthorized access request. Compared to centralized AC model, our proposed scheme has the following advantages:

- *Load balance*: The BlendCAC framework takes advantage of delegation mechanism to distribute the load of centralized PDC server to separate local domain coordinators, such that the bottleneck effect of PDC is mitigated and the risk of malfunction resulting from centralized system is reduced. Even in the worst case when the PDC crash for a short period of time, a large number of domain coordinators still work normally on behalf of the PDC to provide services;
- *Decentralized authorization*: Leveraging the blockchain technique, the proposed BlendCAC scheme allows users to control their devices and resource without depending on a third centralized

authority to establish the trust relationship with unknown nodes; instead, it could define a domain-specific access authorization policy, which is meaningful to the distributive, scalable, heterogeneous and dynamic IoT-based applications;

- *Edge computing driven intelligence*: Thanks to federated delegation mechanism and blockchain technology, the BlendCAC framework provides a device-driven access control strategy that is suitable for the distributed nature of IoT environment. Through transferring power and intelligence from the centralized cloud server to the edge of network, the risk of performance bottleneck and the single point of failure are mitigated, and smart things are capable of protecting their own resource and privacy by enforcing user-defined security mechanism;
- *Fine granularity*: Enforcing access right validation on local service providers empowers those smart devices to decide whether or not to grant access to certain services according to local environmental conditions. Fine-grained access control with lease privilege access principle prevents privilege escalation, even if attacker steals capability token;
- *Lightweight*: Compared to XML-based language for access control, such as XACML, JSON is a lightweight technology that is suitable for resource constrained platforms. Given the experimental results, our JSON based capability token structure introduces small overhead on the general performance.

7. Conclusions

In this paper, we proposed a decentralized federated capability-based access control framework leveraging the smart contract and blockchain technology, called BlendCAC, to handle the challenges in access control strategies for IoT devices. A concept-proof prototype has been built in a physical IoT network environment to verify the feasibility of the proposed BlendCAC. The FCDM model and CapAC policy are transcoded to smart contracts and works on the private Ethereum blockchain network. The desktops and laptops serve as miners to maintain the sanctity of transactions recorded on the blockchain, while Raspberry PI devices act as edge computing nodes to access and to provide IoT-based services. Extensive experimental studies have been conducted and the results are encouraging. It validated that the BlendCAC scheme is able to efficiently and effectively enforce access control authorization and validation in a distributed and trustless IoT network. This work has demonstrated that our proposed BlendCAC framework is a promising approach to provide a scalable, fine-grained and lightweight access control for IoT networks.

While the reported work has shown significant potential, there is still a long way to go to build a complete decentralized security solution for IoT edge computing. Deeper insights are expected. Part of our on-going effort is focused on further exploration of the blockchain based access control scheme in real-world applications. Taking the smart surveillance system as a case study, the proposed BlendCAC will be extended to protect network cameras and motion sensors in the novel urban surveillance platform we developed recently [32,42].

Abbreviations

The following abbreviations are used in this manuscript:

ABAC:	Attribute-based Access Control
AC:	Access Control
ACL:	Access Control List
ACM:	Access Control Matrix
BlendCAC:	BLockchain-ENabled Decentralized Federated Capability-based Access Control
CAC/CapAC:	Capability-based Access Control
FCDM:	Federated Capability-based Delegation Model
IoT:	Internet of Thing
RBAC:	Role-based Access Control

RTT: Round Trip Time
QoS: Quality of Service

1. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials* **2015**, *17*, 2347–2376.
2. Snidaro, L.; Garcia-Herrera, J.; Llinas, J.; Blasch, E. *Context-Enhanced Information Fusion*; Springer, 2016.
3. Blasch, E.; Kadar, I.; Grewe, L.L.; Brooks, R.; Yu, W.; Kwasinski, A.; Thomopoulos, S.; Salerno, J.; Qi, H. Panel summary of cyber-physical systems (CPS) and Internet of Things (IoT) opportunities with information fusion. Signal Processing, Sensor/Information Fusion, and Target Recognition XXVI. International Society for Optics and Photonics, 2017, Vol. 10200, p. 102000O.
4. Alaba, F.A.; Othman, M.; Hashem, I.A.T.; Alotaibi, F. Internet of Things security: A survey. *Journal of Network and Computer Applications* **2017**, *88*, 10–28.
5. Gusmeroli, S.; Piccione, S.; Rotondi, D. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling* **2013**, *58*, 1189–1205.
6. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system **2008**.
7. Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V. Blockchain technology: Beyond bitcoin. *Applied Innovation* **2016**, *2*, 6–10.
8. Ouaddah, A.; Mousannif, H.; Elkalam, A.A.; Ouahman, A.A. Access control in the Internet of things: big challenges and new opportunities. *Computer Networks* **2017**, *112*, 237–262.
9. Gong, L. A secure identity-based capability system. Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on. IEEE, 1989, pp. 56–63.
10. Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E. Role-based access control models. *Computer* **1996**, *29*, 38–47.
11. Samarati, P.; de Vimercati, S.C. Access control: Policies, models, and mechanisms. International School on Foundations of Security Analysis and Design. Springer, 2000, pp. 137–196.
12. De Souza, L.M.S.; Spiess, P.; Guinard, D.; Köhler, M.; Karnouskos, S.; Savio, D. Socrades: A web service based shop floor integration infrastructure. In *The internet of things*; Springer, 2008; pp. 50–67.
13. Spiess, P.; Karnouskos, S.; Guinard, D.; Savio, D.; Baecker, O.; De Souza, L.M.S.; Trifa, V. SOA-based integration of the internet of things in enterprise services. Web Services, 2009. ICWS 2009. IEEE International Conference on. IEEE, 2009, pp. 968–975.
14. Zhang, G.; Tian, J. An extended role based access control model for the Internet of Things. Information Networking and Automation (ICINA), 2010 International Conference on. IEEE, 2010, Vol. 1, pp. V1–319.
15. Yuan, E.; Tong, J. Attributed based access control (ABAC) for web services. Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005.
16. Smari, W.W.; Clemente, P.; Lalande, J.F. An extended attribute based access control model with trust and privacy: Application to a collaborative crisis management system. *Future Generation Computer Systems* **2014**, *31*, 147–168.
17. Ye, N.; Zhu, Y.; Wang, R.c.; Malekian, R.; Qiao-min, L. An efficient authentication and access control scheme for perception layer of internet of things. *Applied Mathematics & Information Sciences* **2014**, *8*, 1617.
18. Liu, B.; Chen, Y.; Hadiks, A.; Blasch, E.; Aved, A.; Shen, D.; Chen, G. Information fusion in a cloud computing era: a systems-level perspective. *IEEE Aerospace and Electronic Systems Magazine* **2014**, *29*, 16–24.
19. Gusmeroli, S.; Piccione, S.; Rotondi, D. IoT@ Work automation middleware system design and architecture. Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on. IEEE, 2012, pp. 1–8.
20. Anggorojati, B.; Mahalle, P.N.; Prasad, N.R.; Prasad, R. Capability-based access control delegation model on the federated IoT network. Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on. IEEE, 2012, pp. 604–608.
21. Skinner, G.D.; others. Cyber security management of access controls in digital ecosystems and distributed environments. 6th International Conference on Information Technology and Applications (ICITA 2009), 2009, pp. 77–82.

22. Hernández-Ramos, J.L.; Jara, A.J.; Marin, L.; Skarmeta, A.F. Distributed capability-based access control for the internet of things. *Journal of Internet Services and Information Security (JISIS)* **2013**, *3*, 1–16.
23. Hernández-Ramos, J.L.; Jara, A.J.; Marín, L.; Skarmeta Gómez, A.F. DCapBAC: embedding authorization logic into smart things through ECC optimizations. *International Journal of Computer Mathematics* **2016**, *93*, 345–366.
24. Swan, M. *Blockchain: Blueprint for a new economy*; " O'Reilly Media, Inc.", 2015.
25. Maesa, D.D.F.; Mori, P.; Ricci, L. Blockchain based access control. IFIP International Conference on Distributed Applications and Interoperable Systems. Springer, 2017, pp. 206–220.
26. Ouaddah, A.; Abou Elkalam, A.; Ait Ouahman, A. FairAccess: a new Blockchain-based access control framework for the Internet of Things. *Security and Communication Networks* **2016**, *9*, 5943–5964.
27. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*.
28. Gomi, H.; Hatakeyama, M.; Hosono, S.; Fujita, S. A delegation framework for federated identity management. Proceedings of the 2005 workshop on Digital identity management. ACM, 2005, pp. 94–103.
29. Aura, T. Distributed access-rights management with delegation certificates. In *Secure Internet Programming*; Springer, 1999; pp. 211–235.
30. Zhang, L.; Ahn, G.J.; Chu, B.T. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security (TISSEC)* **2003**, *6*, 404–441.
31. Chen, N.; Chen, Y.; You, Y.; Ling, H.; Liang, P.; Zimmermann, R. Dynamic urban surveillance video stream processing using fog computing. Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on. IEEE, 2016, pp. 105–112.
32. Chen, N.; Chen, Y.; Blasch, E.; Ling, H.; You, Y.; Ye, X. Enabling Smart Urban Surveillance at The Edge. Smart Cloud (SmartCloud), 2017 IEEE International Conference on. IEEE, 2017, pp. 109–119.
33. Chen, N.; Chen, Y.; Song, S.; Huang, C.T.; Ye, X. Smart Urban Surveillance Using Fog Computing. Edge Computing (SEC), IEEE/ACM Symposium on. IEEE, 2016, pp. 95–96.
34. Xu, R.; Chen, Y.; Blasch, E.; Chen, G. A Federated Capability-based Access Control Mechanism for Internet of Things (IoTs). the SPIE Defense & Commercial Sensing 2018 (DCS), Conference on Sensors and Systems for Space Applications. SPIE, 2018.
35. Ethereum Homestead Documentation. http://www.ethdocs.org/en/latest/index.html.
36. Solidity. http://solidity.readthedocs.io/en/latest/.
37. Truffle. http://truffleframework.com/docs/.
38. Crockford, D. RFC 4627: The application/json media type for javascript object notation (json), July 2006. *Status: INFORMATIONAL*.
39. Flask: A Pyhon Microframework. http://flask.pocoo.org/.
40. SQLite. https://www.sqlite.org/index.html.
41. Go-ethereum. https://ethereum.github.io/go-ethereum/.
42. Xu, R.; Nikouei, S.Y.; Chen, Y.; Song, S.; Polunchenko, A.; Deng, C.; Faughnan, T. Real-Time Human Object Tracking for Smart Surveillance at The Edge. the IEEE International Conference on Communications, Selected Areas in Communications Symposium Smart Cities Track. IEEE, 2018.