OPERATION AND ADMINISTRATION OF ACCESS CONTROL IN IoT ENVIRONMENTS

by

MEHRNOOSH SHAKARAMI, M.Sc.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

COMMITTEE MEMBERS:
Ravi Sandhu, Ph.D., Chair
Dr. Murtuza Jadliwala, Ph.D.
Ram Krishnan, Ph.D.
Palden Lama, Ph.D.
Xiaoyin Wang, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Science
Department of Computer Science
May 2022

DEDICATION

I would like to dedicate this dissertation to the memory of my beloved mother, Mrs. Maryam Khademi, my father, Mr. Mohammadhassan Shakarami, and my sister, Dr. Mehrnaz Shakarami, who have been always my endless and unconditional source of support, inspiration, and love. I would also like to dedicate this dissertation to my beloved husband, Dr. Seyed Ali Mirheidari, whose unfailing love and support sustained me during the most challenging times of this journey.

ACKNOWLWDGEMENTS

First and foremost, I would like to express my profound gratitude to the best Ph.D. advisor one can hope for, Professor Ravi Sandhu, who has been an amazing mentor for me. Through his prodigious knowledge, regular feedback, support, and guidance, as well as his constant pushing of my thinking, he challenged me in all the right ways. Having the opportunity to learn the art of being a better human from such an accomplished scientist is an honor. Professor (as I call him), I will adhere to what I have learned from you to enlighten my path throughout my career and life.

My special appreciation goes out to Mr. James Benson, the adept technology research associate at ICS, who was my great ally to push the hardest obstacles on my research journey. His knowledge and patience were great supports to bring this dissertation to fruition. I would also like to appreciate my committee members, Dr. Ram Krishnan, Dr. Murtuza Jadliwala, Dr. Palden Lama, and Dr. Xiaoyin Wang, who provided me with their valuable feedback, insights, and support to make this dissertation more valuable. I convey my deep gratitude to the staff members from the ICS and the CS department, Suzanne Tanaka and Susan Allen, for their constant support, help, and kindness during my Ph.D.

I would like to dedicate this dissertation to the loving memory of my guardian angel in heaven, my best friend ever, the best mom one could ever imagine, Maryam Khademi. Maman (as I call her), I would never reach this point without you being the endless source of love, support, wisdom, and encouragement in my life. You are my role model and the first person who taught me the value of constant learning and endeavor. This is for you Maman. I also dedicate this dissertation to the best ever dad, my pillar, Mohammadhassan Shakarami. Baba (as I call him), nothing is more satisfying than seeing your smile when I finish this journey. Your words of encouragement and push for tenacity, when I was leaving the country to begin this journey, would always ring in my ears. I am deeply grateful for having your unbounded love and support in all ups and downs in my life.

I would also like to dedicate this dissertation to my loved one, my closet friend, and my husband, Dr. Seyed Ali Mirheidari, who was my inspiration from the beginning to go into the Cyberse-

curity field and was always there for me throughout my doctoral program. I will always remember the times when he sat there for hours for me to challenge my ideas and be my cheerful supporter without whom this journey would not be possible. Furthermore, I am grateful to my adorable, one and only sister, Dr. Mehrnaz Shakarami, who has always been a great source of love and support. I always remember the advice you gave me when you were four and saw me off to final exams, telling me to stay confident. Thanks to you, I feel loved and hopeful until the end of the world.

This Masters Thesis/Recital Document or Doctoral Dissertation was produced in accordance with guidelines which permit the inclusion as part of the Masters Thesis/Recital Document or Doctoral Dissertation the text of an original paper, or papers, submitted for publication. The Masters Thesis/Recital Document or Doctoral Dissertation must still conform to all other requirements explained in the Guide for the Preparation of a Masters Thesis/Recital Document or Doctoral Dissertation at The University of Texas at San Antonio. It must include a comprehensive abstract, a full introduction and literature review, and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported.

It is acceptable for this Masters Thesis/Recital Document or Doctoral Dissertation to include as chapters authentic copies of papers already published, provided these meet type size, margin, and legibility requirements. In such cases, connecting texts, which provide logical bridges between different manuscripts, are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the students contribution to the work and acknowledging the contribution of the other author(s). The signatures of the Supervising Committee which precede all other material in the Masters Thesis/Recital Document or Doctoral Dissertation attest to the accuracy of this statement.

May 2022

OPERATION AND ADMINISTRATION OF ACCESS CONTROL IN IoT ENVIRONMENTS

Mehrnoosh Shakarami, Ph.D.
The University of Texas at San Antonio, 2022

Supervising Professor: Ravi Sandhu, Ph.D.

The Internet of Things (IoT) denotes a network of evolving and expanding number of technologies embedded in smart *things* with at least one network interface to interact with the physical and digital world. IoT has gained widespread use cases in the market, ranging from individual customers implementing IoT in their personal homes to industry and organizational customers utilizing IoT in their business environment. The reason for this widespread popularity of IoT in different application domains include convenience, automation, energy efficiency, and other functionalities which IoT brings to different environments. However, many IoT customers are not aware of potential direct or indirect security hazards to which they or their environment might be posed by utilizing unsecured IoT.

The availability and efficiency of security capabilities for IoT environment is often different from conventional IT environments, because of IoT unique characteristics, including being used in dynamic environments, limitation in power and computational resources, and relying on heterogeneous configurable firmware/platforms. Therefore, providing appropriate security mechanisms for IoT application environments gained momentum in both academic and industry communities. One of the most important security concerns in IoT is access control which is still open to novel and effective solutions. In order to design an appropriate access control approach for an IoT environment, the distinct specification and requirements of that environment have to be considered. Although general requirements for designing appropriate access control solutions for IoT applications have been stated in the literature as being scalable, dynamic, interoperable, context-aware, fine-gained, etc., these requirements may be of different priorities for different IoT environments. Therefore, a single access control solution cannot cater to various IoT applications because of their

different requirements and characteristics.

In this dissertation, we focus on smart home IoT as a prevailing IoT application domain which has unique characteristics. Home IoT environment may include different IoT devices shared among different users. There are complex social relationships among home IoT users, including parents, kids, babysitters, visitors, etc., which offers different threat models. Moreover, a smart home environment might be extended over time through adding new IoT devices to the house by the homeowner. Home IoT devices are probably produced by different vendors, therefore relying on different platforms. Nevertheless, we need different home IoT devices to be interoperable to facilitate home automation. In terms of access control requirements, some of them are of more importance than others. As an example, it is more essential for an control solution in the smart home to be fine-grained rather than being scalable. These especial characteristics and authorization requirements call for tailored access control solutions to be designed for smart home IoT environments. Surprisingly, little attention has been paid to access control specification in smart home IoT.

In this dissertation, we investigate three major access control-related topics which affect or directly provide authorization in the home IoT environment. First topic is concerned with the problem of *inconsistency* which is defined as provision of outdated authorization information to the decision point which may lead to access violation. Due to intermittent Internet connection and limited storage space of home IoT devices, required authorization information might not be available in real time. So, there is an increased risk of making access control decisions based on outdated information. This problem may arise in any attribute-based access control environment in which attributes are provided incrementally to the decision point. We investigate this problem in general, interpreted with use cases in a smart home IoT environment.

Another overlooked area in smart home IoT environments is administration of access while overall system security is crucially dependent on both administrative and operational authorization models. Since home users are usually not IT experts and are less likely to spend time to learn complex management interfaces, administration of access in smart home IoT turns is particularly

problematic. In this dissertation, we propose an administrative access control model for a smart home IoT considering its specific dynamics and characteristics, which is backed up by a proof-of-concept implementation.

Finally, we take a first step towards addressing one of the most unique and novel requisites toward realization of smart home IoT automation, which has received surprisingly little attention so far. A holistic view of home automation demands specific access control specifications to facilitate inter-device interactions. In this dissertation, we propose a novel authorization framework based on attribute-based access control which includes access control model specification, enforcement architecture and a proof-of-concept implementation. The proposed model is designed to regulate device-to-device inter-communications in a smart home IoT environment. We regard our solution to be a first step towards providing more comprehensive access control approaches pertinent to the interoperable IoT requirements.

Some future directions and research agenda are discussed in the conclusions of this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: INTRODUCTION

Internet of things (IoT) is a trending technology of connected embedded objects, a.k.a things, that can sense, communicate, actuate and be accessed through a communication channel. In 2011, IoT was identified in the Gartner hype cycle, which represents the emergence, peak and adoption of innovative technologies [12]. IoT paradigm as a collective environment of cyber-physical devices, people and applications gained ground as a technology in which information exchange happens to provide services. This hyper-connected digital ecosystem is quickly becoming a reality and is gradually changing people's everyday lives. While connected IoT environments benefit individuals, society and industry through facilitating automation, convenience and efficiency, they also bring new security and privacy challenges. These challenges are exacerbated by lack of standards specifically designed for IoT devices which are typically resource-constrained and rely on heterogeneous technologies [80].

According to OWASP IoT Project [14], insecure access to web, APIs, and mobile interfaces are among top-10 IoT security vulnerabilities. Nonetheless, access control is a missing component in many commercial IoT frameworks. Most IoT frameworks enforce coarse-grained access control policies, for example, Nest thermostat grants access to all available functionalities of the device or none. Another example is Apple Home Kit, which provides view-only, local, or remote control as the access control options. Samsung SmartThings has slightly finer-grained access policies to access IoT services, however it needs the location of users to be revealed, which violates users' privacy. There are numerous examples of inappropriate access control for different IoT devices, which lead to safety and privacy violation of their users. Remote hacking and controlling of Internet-based baby monitoring systems in 2015 [191] is just one example which resulted from over-privileged access granted by its authorization framework. In this context, devising and enforcing appropriate authorization frameworks for IoT environments is of utmost importance to address current authorization frameworks limitations.

Different IoT-enabled smart environments include smart cities, smart homes, smart grid, smart

buildings, smart health, smart transport and smart industry [181]. Each IoT smart environment has its specific consumer group with different interests, priorities and regulations depending on the criticality of provided functionalities by IoT devices in that environment [130]. Similarly, the key requirements of scalability, interoperability, availability, performance and reliability, and their attendant tradeoffs, are different in each smart environment. Each environment has different characteristics, therefore different requirements. For example scalability is more important in a smart transportation system to which many vehicles may join at any time. Performance, including minimizing the communication and computation overheads, is critical for applications like patient monitoring and smart cars. As another example, an adequate level of reliability and availability is critical in a smart health environment, while occasional unavailability/failure might be tolerable in a smart home [160]. On the other hand, interoperability as seamless integration of heterogeneous IoT devices, is remarkable for all IoT applications. Therefore, a single solution cannot cater to IoT applications in various fields due to different requirements of each. In a similar way, each IoT environment demands appropriate access control framework designed to address its particular requirements. In this research, we studied three major access control-related matters which affect or directly provide authorization in IoT environments.

First, we investigate the safety and consistency issues in distributed Attribute-Based Access Control (ABAC) environments. ABAC is one of the most popular access control approaches for designing IoT access control models due to its flexibility, granularity, expressiveness and context-awareness. ABAC relies on attribute values of identities, including subjects, objects, environment (context) and actions, to define access control policies. Attribute values are represented in credentials which have been properly bound to their holders' identities. However, credentials might be updated or revoked during their lifetime. So, if any access decision would be made based on outdated attribute values in changed/revoked credentials, it would be incorrect. We call this *safety and consistency problem*. This problem may arise in any distributed ABAC environment in which attribute values are provided through multiple, distributed authorities. In an IoT environment, with dynamicity as one of its inherent characteristics, these changes might be frequent. Hence, any

changes in identities' attributes should be propagated in a timely manner to avoid incorrect access penalties which may happen as a result of safety and consistency problem. To serve this purpose we formalize different consistency levels for attributes in an ABAC environment to mitigate this problem as far as possible.

Second, we narrow our focus to smart home IoT. As previously stated, each IoT environment has its own particular requirements for access control. We choose smart home IoT as our scope in this dissertation due to its important role in people's everyday lives which is growing, it being susceptible to specific security vulnerabilities due to its distinctive features, and being less investigated compared to other IoT application domains. We present an administrative Role-Based Access Control (RBAC) model to manage access authorizations in a smart home IoT environment. We recognize access administration in a smart home to be a particular problem as home users lack the expertise of a typical system administrator and are unlikely to spend much time learning complex interfaces to assign/revoke access rights or auditing the access logs.

Finally, we propose a decentralized, ledger-based architecture along with a proof-of-concept implementation on Ethereum blockchain using smart contracts. Our implementation results demonstrate that using blockchain for administrative access control is viable, while is not yet appropriate for operational access control, despite the rising hype around blockchain technology and its utilization in the access control domain. Although our selected underlying operational access control is a RBAC model, our proposed administrative model could be utilized for access management of any underlying operational model, regardless of it being RBAC or used any other access control paradigm. Moreover, the proposed model could be simply extended to manage more sophisticated access control models with similar dynamics, e.g., smart buildings.

Since IoT has been envisioned as a network of connected IoT devices, device-to-device communication is essential to realize full benefits of the IoT ecosystem. There are situations in which co-located devices need to interoperate using heterogeneous communication technologies, which makes it challenging. Extensive efforts are going on to provide direct intelligent communication among IoT devices, however there is still no standard or consensus on any paradigm. Access con-

trol as the backbone for any information system, is of utmost importance to be addressed in order to facilitate device-to-device communications. Surprisingly there is no previous access control specification that was provided to regulate device-to-device access. Third proposal in this research is to propose an access control model for the first time for regulating device-to-device access in the smart home IoT environment. We formulate an access control model which governs authorized flow of information among home IoT devices using Attribute-Based Access Control (ABAC). Our approach relies on the message-passing paradigm for regulating access in device-to-device interactions. Furthermore, we introduce the concept of *scenarios* which reflects a chain of events in the smart home context. We develop a scenario-driven attribute-based access control which also enables conflict resolution via defining partially ordered sets of scenarios using *priorities*. The viability of the proposed approach is substantiated via a formal model and an enforcement architecture, backed up by a proof-of-concept implementation which affirms our model is quite efficient.

1.1 Motivation

Similar to any information system, IoT environments need to ensure the security and privacy of data and resources. Due to the dynamic nature of IoT environments, in which users, IoT devices, and IoT applications are actively interacting with each other, it calls for tailored security and privacy solutions to provide required privacy and security. Different application domains and environments demand for different access control models and mechanisms. Similarly, IoT applications and environments which are heterogeneous, dynamic and resource-constrained and also sometimes demand for decentralization, privacy and auditability, will ask for tailored security mechanisms to tackle their specific requirements. Access control, as the backbone of information security, can benefit IoT environments via actively monitoring access to resources and protecting them against unauthorized access. This research deals with a very specific application of IoT—the smart home IoT.

According to Statista research, the number of active households in the smart home market would exceed 84 million by 2026 [17]. While bringing convenience to people's lives, usage and

application of IoT in a smart home, a.k.a home automation, introduces new security challenges. There is a growing research on individual parts of smart home components. For instance, there are studies on vulnerabilities of communication protocols [78, 166], device firmware security [95, 122], and home automation applications [47, 74, 118]. In this research we focus on securing smart home IoT through provision of required authorization settings, specifically access control models, which is surprisingly not intensively investigated [91]. We recognize smart home IoT unique characteristics necessitate oriented authorization models to be particularly designed. These unique characteristics include diversity and heterogeneity of utilized IoT devices, mixed ownership of devices, possible conflicts of interests, and lack of a dedicated expert administrator [108].

Attribute-Based Access Control (ABAC) is a widely adopted model in IoT environments because of its flexibility and expressiveness. Incremental assembly of required subject attributes with different validity periods creates potential for inconsistency leading to incorrect access decisions. Violation by relying on outdated validation information is a common problem in access control enforcement. Following example showcases how consistency problem in a smart home environment may lead to access violation. Consider a smart lock with an access control system which is deployed on the vendor's private cloud. Smart lock would authenticate and pair with a user's mobile in its Bluetooth range, then receive the status of that specific user's access key from the cloud. This key status would be also saved in the local database of the lock. Now imagine a scenario in which a user's key has been revoked by the owner. This change would be reflected in the local database of the smart lock, only when that specific user interacts with the lock through his/her smartphone. A malicious user whose key has been revoked by the owner, can disconnect her phone from the Internet and get into Bluetooth range of the lock. Since the lock cannot connect to its cloud through the user's phone, it would rely on the user's key information stored in its local database and grant access to that user, while it is supposed to deny it. This problem arises because the smart lock is relying on outdated keys, which we call *safety and consistency problem*. This situation has been referred to as *state consistency attack* in [97]. Similar access violations may happen due to provision of stale/outdated attributes to the decision point when access decision is

made based on provided keys/attribute certificates. We investigate this problem in detail in this dissertation and propose different levels of consistency to provide the decision point with the most recent status of subjects' attributes. To avoid safety and consistency problem, which is caused by relying on expired/revoked credentials, we proposed multiple increasingly stringent consistency specifications, along with more detailed smart home IoT use cases.

The other challenge in the area of IoT access control is about designing appropriate access control models. Despite many efforts which have been done toward proposing access control methods for IoT environments in general [157, 160], there are only a few works which propose access control *models*. Specifically, smart home IoT as one of the most popular IoT application domains has been surprisingly overlooked [214]. Besides, the few proposals in this area are mostly concentrated on operational access control models [24]. However, the overall system's security mainly depends on both operational and administrative access control models. Administrative models facilitate management of (mostly configuration) changes in the underlying operational models. In this dissertation, we have developed a RBAC administrative model for the smart home IoT environment, with its enforcement architecture relying on blockchain. The model has been backed up by a smart home IoT proof-of-concept implementation.

Last but not least, we noticed existing studies are mostly focused on permission models and authorization in user-centric modes [24, 74, 118] and concerned with user-to-device communications, protecting IoT devices from unauthorized users' access. Nonetheless, the potential security violations which may happen in device-to-device interactions are largely uninvestigated [218]. In order to realize the vision of IoT, as a network of connected smart things, it is required to consider the environments with ongoing device-to-device communications, without human intervention. Smart home IoT is no exception and scenarios of device-to-device interactions are inevitable for real-life home automation. To this end, we embark on proposing the first device-to-device access control model in a smart home IoT environment. In this dissertation, we propose a novel device-to-device access control model and its extension to scenario-driven attribute-based access control for device-to-device interactions in a smart home IoT. Our approach relies on message passing as the paradigm

for device-to-device interactions. Formal specification of the proposed models, along with smart home use case scenarios, its enforcement architecture and a proof-of-concept implementation have also been provided.

1.2 Problem Statement and Solution Approach

There is a rich body of research on access control in different IoT application domains, as unauthorized access to the information flow in IoT environment may impose critical security issues in those environments [1, 37, 133, 148, 157, 160]. In a smart home environment, as the area of our focus, there are a few access control models proposed based on RBAC [24] or ABAC [26], while some researchers believe a hybrid approach would be the most appropriate to govern the access in a smart home [26]. Many researchers considered ABAC to be more appropriate for generation of access control policy in IoT environments, however less attention has been paid to the consistency problem which may arise in ABAC environments due to relying on credentials for attribute value provisions. Another overlooked area is the access management regardless of the access control model in the underlying operational environment. Most importantly, there is no effort in the access control model provisioning access in device-to-device communication. Nonetheless, the internet of connected heterogeneous devices (IoT) would remain a vision rather than a reality until this important area remains disregarded.

In order to investigate above-mentioned problems in access control for smart home IoT we first look into the consistency problem, in order to provide a formal analysis of the problem and provide administrators with different levels of safety and consistency which could be acquired in each level. Then we will focus on administration of access in the smart home IoT environment. We consider operational models to be dynamic as we are aiming for a smart home IoT environment and recognize the need to develop administrative models in order to govern access changes in such a system. A RBAC administrative model will be provided to address this need with a smart home use case, however scalability and flexibility to be used in other distributed environments has been taken into account. To provide a holistic perspective toward home automation, we consider

heterogeneous IoT devices at home as an ecosystem which demands for specific device-to-device access control models to be designed for it. We will propose an access control model which is proposed to regulate the access and authorize communications among IoT devices in a smart home environment.

1.3 Thesis Statements

1. The established paradigm of role-based access control can be utilized for access **administration** of user-to-device in corresponding operational access control models, which could be based on either role-based or attribute-based access control.

2. If the operational authorization is based on attribute-based access control, a detailed analysis of required **consistency** for both mutable and immutable attribute values can ultimately benefit the overall safety of the system by providing a decision point with most recent values of attribute credentials.

3. The established paradigm of attribute-based access control can be adapted and extended to provide a **device-to-device** access control approach towards considering heterogeneous IoT devices in a smart home as an ecosystem with intercommunication.

1.4 Scope and Assumptions

The scope of this dissertation is to first investigate safety and consistency problem in distributed attribute-based access control environments, for both mutable and immutable attributes. We propose using refresh instead of revocation check to be applied, specially for mutable attributes. Then a role-based access control administrative model is introduced for managing assignments in the underlying operational model for a smart home IoT. Then a novel device-to-device attribute-based access control model is proposed for the first time to regulate device inter-communications in an automated smart home IoT. Followings are assumptions made in different sections of this dissertation:

1. We examine the safety and consistency problem from the perspective of a single access decision point within a larger distributed attribute-based access control authorization system. We assume required credentials for evaluating a policy would be collected incrementally and lifetimes of different credentials might not be the same.

2. In attribute-based access control, the value of an attribute of a subject is represented by a credential which must be coupled to a specific subject, which is typically achieved by embedding the subject's identity in the credential. We assume the identity of the subject was authenticated before the credential is coupled with that subject.

3. About consistency levels in attribute-based access control, we assume that the policy and object/environment attributes are known with high assurance at the decision point, which is reasonable since the decision and enforcement points are typically co-located with the object's custodian who maintains these values. This reduces the problem to safety and consistency of *subject* attributes.

4. In order to achieve consistency levels for mutable attributes, we utilize a quota-based approach and assume each mutable attribute has a global limit known to its corresponding attribute authority and can be managed in a centralized way.

5. For administration of access, our model specifically addresses the administration of EGRBAC [24]. However, it could be simply extended to manage other more sophisticated access control models with similar dynamics. EGRBAC is chosen not as a de-facto operational model, but because it has the desired properties for a smart home IoT operational access control.

6. Our proposed administrative model is restricted to manage user-to-device assignments. We recognize adding a new user to be an infrequent event. So, we consider this case orthogonal to the central focus of this research and its administration would be centralized, say, in the homeowner.

7. For proposing a device-to-device access control model, as many IoT devices are not IP-enabled, using a gateway (GW) node in the network is inevitable [34]. We assume the gateway node in our model is trustworthy and available, which is a common assumption [156]. Attacker is considered to be an outsider to the network with the goal of obtaining illegitimate access to available functionalities/operations of smart home IoT devices.

1.5 Summary of Contributions

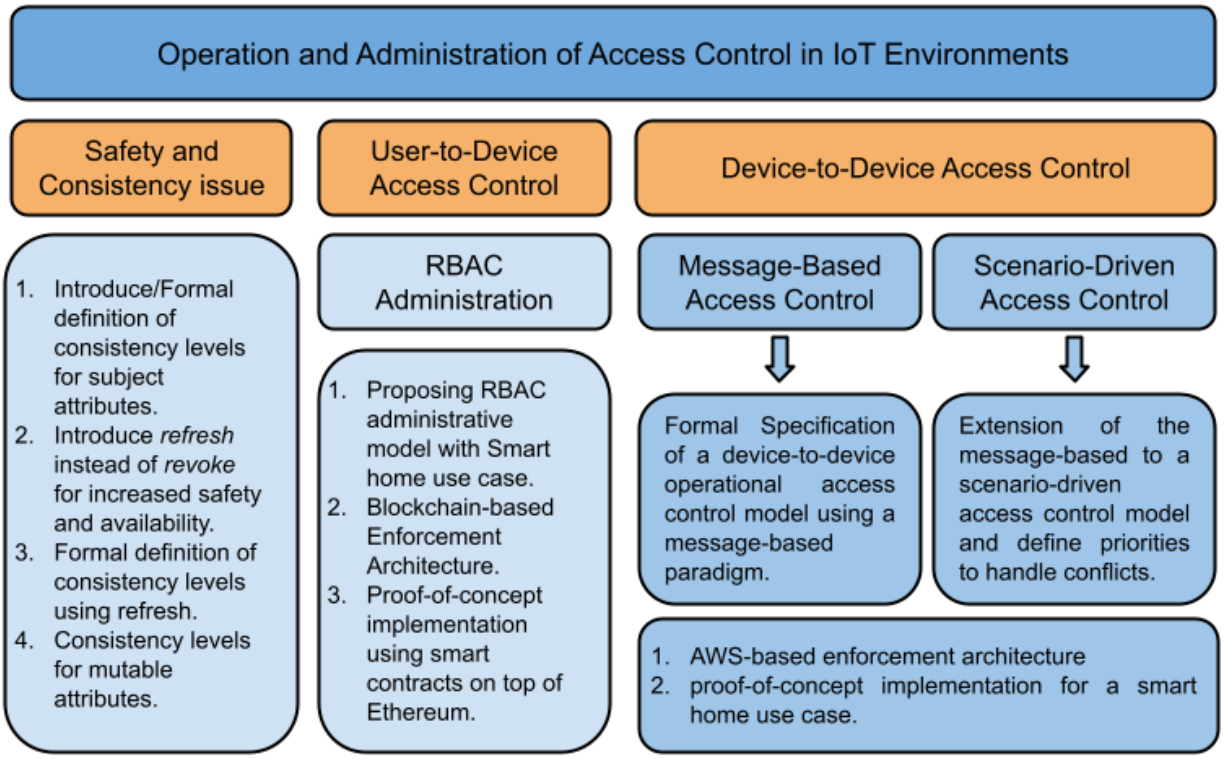

Figure 1.1: Overview of Contributions

Figure 1.1 depicts an overview of contributions in this dissertation, which are described as follows:

- To address safety and consistency problem in distributed ABAC environments, we propose a set of five consistency levels in this research which are partially ordered in increasing strictness. Formal specification of each of our consistency levels is defined and we identify the properties guaranteed by each level. Implications of proposed consistency levels in a

smart home IoT environment have been provided as well.

- We develop a formal framework for safety, availability and consistency problems of ABAC systems via introducing the refresh scenario instead of the traditional revocation check. The enhanced possibility of getting a new value rather than an invalid response enhances safety and availability. We also define the concept of being *satisfactory* for an attribute value with respect to a policy, which is first introduced in our work to the best of our knowledge. Relying on the history of satisfactory attribute values, we introduce additional flexibility to grant access to authorized users. A smart home IoT use case is also provided.

- Safety and consistency in the context of mutable subject attributes is also investigated in this research. We develop a formal characterization of required consistency using refresh in this context. This work has also been consolidated with a smart home IoT use case.

- In this dissertation, we have developed a corresponding RBAC administrative model for EGRBAC [24] (enhanced generalized role-based access control), the access control model which we chose at the operational level of access control in a smart home IoT.

- We take advantage of blockchain to enforce our proposed administrative model, therefore make it reliable, auditable, and scalable, so it could be scaled for administration in environments with similar dynamics, e.g. smart buildings. We opted for Ethereum blockchain and designed the policy enforcement model and implemented it on top of Ethereum. Our implementation results are reassuring that although the use of blockchain for operational access control is not promising, despite the hype around using blockchain for operational access control, an administrative model could successfully utilize the benefits of blockchain.

- An access control model which governs authorized flow of information among home IoT devices using Attribute-Based Access Control (ABAC) has been proposed for the first time, relying on a message passing paradigm.

- We define *scenarios* as a sequence of messages which is provoked by a trigger, i.e., an event

or a set of events in the smart home. By defining priorities among scenarios, we equipped our proposed model with conflict resolution. Proposed model is backed up by an enforcement architecture and a proof-of-concept implementation for a smart home IoT use case.

1.6 Organization of Dissertation

The rest of this dissertation is organized as follows. A brief background on safety and consistency problem in attribute-based access control environments, access control models for user-to-device access control models, as well as related works in the context of device-to-device communications are discussed in Chapter 2. Chapter 3 presents proposed consistency levels in distributed attribute-based access control environments. Using refresh instead of conventional revocation check and its provision of extra safety and availability has also been discussed. Consistency of the subject's mutable attributes is also deliberated in this chapter. All proposed levels of consistency are followed by their guaranteed properties and reinforced by smart home use cases. In Chapter 4, we discuss our proposed administrative model along with its blockchain-based enforcement architecture in. It further demonstrates the proposed model using a use-case diagram for a smart home use case and a proof-of-concept implementation. Models characteristics and limitations along with safety considerations are also discussed. Chapter 5 discusses the novel proposed approach for device-to-device access control which is an attribute-based access control model relying on a message-passing paradigm. It also describes the extension of the proposed model to the scenario-driven access control which handles conflicts in an automated smart home. Properties of the proposed approach as well as its restrictions are also discussed. This dissertation is concluded in Chapter 6 which summarizes this dissertation's contributions and discusses future directions of work.

Chapter 2: BACKGROUND AND LITERATURE REVIEW

This chapter provides a brief review of the related literature and discusses mostly related works which are essential to comprehend the rest of this dissertation. We first review access control models which are more adopted in IoT environments including RBAC and ABAC. EGRBAC is then discussed as the operational model of our choice for operational access control in smart home IoT. We also provide a brief overview of blockchain concepts and its application in IoT access control. Literature review of research works on device-to-device communications is also presented. Sections of this chapter are arranged based on the sequence of related dissertation chapters.

2.1 Safety and Consistency Problem

Beyond the need to keep the data consistent in open and distributed systems, which has been discussed in the literature (see for example [19, 36, 88, 154, 192]), there is a crucial requirement to have access control models relying on the most recent information to grant/deny access to that data [110]. Many access control models are not completely compatible with distributed systems in that they are not deployed for such systems in the first place [100]. ABAC is well adjusted to distributed environments due to its flexibility and granularity. ABAC determines access based on attributes of subjects, objects and environment evaluated against a policy. These attributes and also the policy are exposed to change and staleness during the time, which could result in inconsistency.

Ciphertext-Policy Attribute-Based Encryption [48, 49] is broadly applicable in decentralized multi-authority environments, but presents challenge to handle attribute revocation [176,208–210]. Moreover, it imposes a heavy performance burden which makes it impractical [81]. On the other hand, delays and staleness of attributes are inherent in every distributed system owing to network latencies, caching and failures [112]. So, most practical distributed systems try to have a near-consistent [110] property, which could be interpreted as limiting the exposure of access control models to stale attributes.

Lee and Winslett propose the first organized work focused on consistency issue in trust nego-

tiation, which is described in more details in following section [116]. In another research work on conversational web services [149], authors build their access control model based on user's credentials which relies on the first, most permissive level of consistency introduced in [116]. Authors simply assume the validity of a credential will last for the whole web service conversation duration. This, however, may result in granting access based on outdated credentials' states. Squicciarini et al. [190], present a protocol which safely performs trust negotiation during distinct negotiation sessions. Even though the authors put the probability of expired/revoked credentials during negotiation suspensions under consideration, they only mention that a synchronization algorithm would take care about updating the list of credentials without describing a concrete underlying synchronization scheme.

There is another category of research work which considers policy changes as the main concern. In [81,103,212], authors concentrated specifically on policy consistency in dynamic environments. They define different variants of proofs of authorizations based on the time when different queries are checked against policies in a transaction lifetime in a cloud environment. In our research in Chapter 3, we consider policy inconsistency to be out of scope and assume that policy is known with high assurance at the decision point. This is similar to other research works [117] in which consistent policy is simply considered as an underlying assumption.

Another closely related research is [111], in which the authors formally specify a set of attributes in linear temporal logic in a Group-based Secure Information Sharing (g-SIS) model to express freshness of attributes. By proposing different levels of stale-safe property, authors try to limit unsafe access decisions made based on stale subjects' and objects' attributes in a distributed access control system. This research is very close to the work in [116] in that it tries to regulate some levels of safety against stale-attribute usage, but with a key difference: authors in [111] built their g-SIS system based on the assumption that the system should be able to cope with loss of communication and limit the exposure to the risk only by relying on the latest time the attributes have known to be valid (refresh time), while in [116], authors focus on the need to obtain the fresh information about credentials' states via revocation checks.

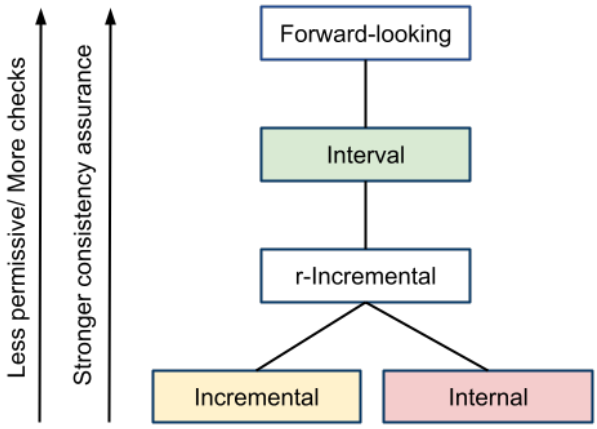

Figure 2.1: Lee-Winslett Proposed Consistency Levels

Our proposed approach in Chapter 3 differs from fail-secure access control models [199] in that we assume that we can acquire fresh revocation status of credentials, but fail-secure access control applies in scenarios in which revocation states cannot be updated or accessed. Our main concern is, although we checked credentials' revocation status, provided information might become obsolete since the last status update. Although our approach depends on revocation status of credentials, being online is not central to our scheme. We provide different interpretations of proposed levels in Chapter 3 in scenarios like short-lived credentials where the only possible revocation check time is the start time of the certificate or credential.

2.1.1 Lee-Winslett Safety and Consistency in Trust Negotiation Systems

The closest research to our work is presented in [116] by Lee and Winslett (hereafter we call it LW), in which authors tried to limit the unexpected and unsafe behavior of trust negotiation and distributed proving authorization systems by proposing four levels of consistency. Trust negotiation systems are a specific type of distributed proof systems [114] which are appropriate when privacy is a major concern [151]. In trust negotiation systems, both parties need to incrementally establish trust to achieve desired degree of assurance, while revealing no more than necessary information. So, collection of credentials would inevitably span over time. The authors considered each credential as a piece of evidence needed to satisfy a given policy, collected over a non-instantaneous period of time. They considered credentials as small snapshots of the whole

network state which cannot be considered a precise snapshot of the global network. Although trust negotiation is not a prerequisite for disclosure of attributes in our proposals in Chapter 3, [116] is the closest work to us.

LW provides notions of consistency levels leveraging the temporal constraints on timing and sequencing of checks for certificate issuance/revocation and demonstrate how failure in satisfaction of these constraints could violate the basic safety requirements of the system. We recast the work of [116] by eliminating the negotiation aspects and developing different levels of consistency with strict subset property among levels. The authors define an entity's view of the system based on collected credentials' states and propose four levels of view consistency including *incremental*, *internal*, *endpoint* and *interval* consistency as presented in Figure 2.1. The first and most permissive level provides *incrementally* consistent view, in which it is sufficient to present valid credentials to satisfy each clause of the policy. The proposed credentials should have been valid at the receive time.

Incremental consistency does not guarantee simultaneous validity of different credentials and it, also, might be in conflict with separation of duties principle. So, the authors propose the second level of consistency which is called *internal*. Internal consistency guarantees all relevant credentials to the authorization decision were valid simultaneously at some point in time during the authorization protocol, but not at the moment the policy is decided to be satisfied. To afford the policy evaluator with a stronger guarantee about authorization decision, authors provide two stronger consistency levels: *endpoint* and *internal* consistency. These two levels provide a formal assurance that not only credentials are all valid at some point during authorization, but also they are all valid at the endpoint of authorization protocol.

Lee and Winslett extend the proof construction in authorization systems to the context-sensitive environments in which parts of the proof tree are required to remain hidden due to privacy concerns [115]. The proposed framework relies on signed assertions instead of certificates issued by certificate authorities (CAs). The authors show how a distributed proof system can sample the simultaneous truth of facts while conforming to integrity and confidentiality policies.

2.2 IoT Access Control Models

IoT application contains its utilization in different environments containing home, industry, health, etc. Despite increasing adoption and tendency of customers to use connected IoT applications, in which different IoT devices can connect and communicate, yet the comfortable features of an interconnected environment, comes with security challenges. There is a rich body of research on security of IoT [27, 29, 90, 131, 211]. Access control, as a backbone for ensuring security in any information system, has gained a lot of attention in IoT applications [1, 37, 133, 148, 158, 160]. Unauthorized access to the information flow in IoT environments may impose critical safety and privacy issues to IoT users.

Besides specific requirements for access control which is common in all IoT application domains, such as fine granularity, context-awareness, dynamicity and lightweight-ness, each requirement has a different level of importance depending on the application domain in which IoT has been utilized. As an example, scalability is much more important in industrial IoT than smart home IoT [160]. As one of the most popular domains of IoT applications, smart home IoT demands for specific access control models tailored for it [58, 92, 106, 193, 202, 214].

A context-sensitive access control approach for a smart home has been proposed in [69] in which policies are focused to control access to users' personally identifiable information (PII). Authors use semantic network knowledge graphs to define the context in a smart home environment and supplement their work with an anomaly detection sub-system to inform users about suspicious activities. Another related research is reported in [38] in which stand-alone ABAC model was proposed for smart home environments, considering the NIST Next Generation Access Control (NGAC) [75] specifications to specify ABAC requirements. Both of these works lack in presenting a specific operational model for their proposed approaches. Ruledger [72] is a ledger-based framework which considers D2D interactions but the goal is to ensure rule integrity in trigger-action IoT platforms, not access control.

Some researchers relied on Attribute-Based Access Control (ABAC) paradigm because of provided flexibility and granularity, suggesting authorization models to govern user to device access

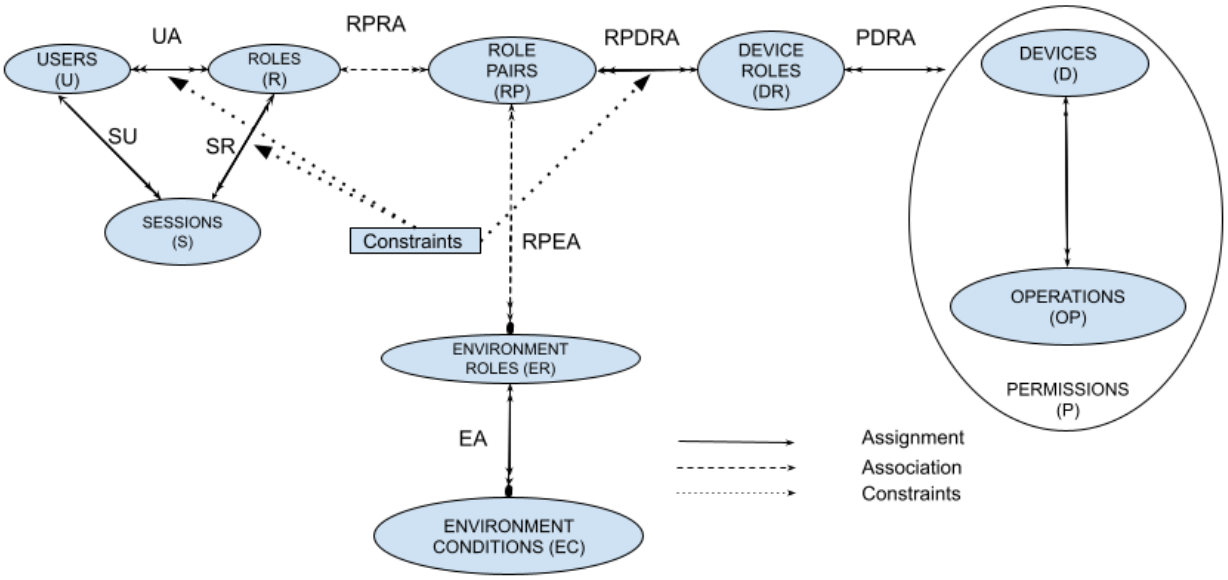

Figure 2.2: Adopted Operational Access Control Model for Smart Home IoT (EGRBAC) [24]

in smart home applications [25, 26, 67]. Bakir et al. [31] proposed a capability-based authorization scheme, namely CAPLet, utilizing Capability-Based Access Control for resource-restricted IoT deployments. Another group of researchers rely on Role-Based Access Control (RBAC), which has been reported the easiest for home users to adopt with [187], and easier to manage [22, 214]. EGRBAC [24] is an RBAC fine-grained model for authorization of user-to-device communications in smart home, which we adopt in this research as the operational model, based on which we build an administrative framework (including policy model, enforcement architecture and a proof-of-concept implementation). This model is briefly described as follows.

2.2.1 EGRBAC: Extended Generalized RBAC for Smart Home IoT

EGRBAC has been proposed by Ameer et. al. [24], to provide a fine-grained access control model for smart home environments. EGRBAC takes into account requirements and challenges of the access control in a smart home environment and enhances over traditional RBAC in order to satisfy the required properties. These characteristics along with a formal model proposed in the work inspired us to consider it as our operational model of choice. Authors provide a finer grained RBAC model, compared to existing models, in that the scope of control has been defined to be at

device-operation level. Instead, other RBAC models in the same context commonly provide the device level granularity of control. Even with one IoT device in the smart home, some dynamics are essential to be considered. It is quite possible for access objects to be added/deleted to the smart home environment. Moreover, in many situations it is a requisite to provide permission-level access instead of it being at device level [105, 194]. A device (D) could be any smart home IoT device like smart security camera, etc. An operation (OP) is an action specified by the manufacturer, which could be done by an IoT device, such as recording. Each permission (P)in EGRBAC is a subset of $D \times OP$. Different Device Roles (DR) have been created based on categorizing available manufacturer-specified operations in a device. It is also possible to put pairs of (device, operation) in the same (DR) for different devices. Then, permissions would be assigned to device roles instead of devices themselves, making the model permission-centric. As a result, it is possible in EGRBAC to grant partial access to a device for different users, for instance a DRcalled Dangerous Devices could contain on/off operation for the oven as well as turning smoke detector on/off. It is possible to delimit user's permission in the system based on their specific permissions on determined device roles.

There are many situations in which it would not be reasonable to make access control decisions only based on roles assigned to individuals. Instead, other contextual information such as environmental conditions and location should be involved in access control. EGRBAC captures environmental context such as time and location using Environment Conditions (EC) which subsequently would activate/deactivate Environment Roles (ER). For instance, light sensors would capture the daylight and determine whether it is daytime or nighttime. Multiple subsets of ECs could be grouped together as an ER, which would later be coupled by regular roles to create Role Pairs (RP). EGRBAC assigns Device Roles (DR) to Role Pairs (RP) to establish the access policy, by defining $RPDRA$ relationship. EGRBAC operational model has been depicted in Figure 2.2. Formal definition of EGRBAC is given in Table 2.1.

In EGRBAC, a user could be assigned to a subset of roles. Then, based on the active roles in his/her current session and the current environmental context, corresponding role pairs would

Table 2.1: EGRBAC Model Formalization [24]

Users, Roles and Sessions
$-U, R$ and S are sets of users, roles and sessions respectively
$-UA \subseteq U \times R$, many to many users to role assignment (homeowner specified)
$-SU \subseteq S \times U$, many to one sessions to user relation that assigns each session to a single user who controls the session
$-SR \subseteq S \times R$, many to many session to roles relation that assigns each session to a set of roles that can change under user control, where $(s_i, r_j) \in SR \Rightarrow (\exists u_k \in U)[(s_i, u_k) \in SU \wedge (u_k, r_j) \in UA]$; by definition of SU, u_k must be unique

Devices, Operations, Permissions and Device Roles
$-D, OP, P$ and DR are sets of devices, operations, permissions and device roles respectively
$-P \subseteq D \times OP$, every permission is a device, operation pair (device manufacturer specified)
$-PDRA \subseteq P \times DR$, a many to many permissions to device roles assignment (homeowner specified)

Environment Roles and Environment Conditions
$-ER$ and EC are sets of environment roles and environment conditions respectively
$-EA \subseteq 2^{EC} \times ER$, many to many subsets of environment conditions to environment roles assignment (homeowner specified)

Role Pairs
$-RP \subseteq R \times 2^{ER}$, a set of role pairs specifying all permissible combinations of a user role and subsets of environment roles (homeowner specified);
for every $rp = (r_i, ER_j) \in RP$, let $rp.r = r_i$ and $rp.ER = ER_j$
$-RPRA \subseteq RP \times R$, many to one role pairs to role association induced by RP, where $RPRA = \{(rp_m, r_n) \mid rp_m \in RP \wedge rp_m.r = r_n\}$
$-RPEA \subseteq RP \times 2^{ER}$, many to one environment roles to role pairs association induced by RP, where $RPEA = \{(rp_m, ER_n) \mid rp_m \in RP \wedge ER_n = rp_m.ER\}$

Role Pair Assignment
$-RPDRA \subseteq RP \times DR$, many to many role pairs to device roles assignment (homeowner specified)

Check Access Predicate
$-$ The check access predicate takes four inputs: session s_i, device d_j, operation op_k and set of active environment conditions EC_l; a session s_i can access device d_j with operation op_k when the set of environment conditions EC_l is active iff the following predicate is true:
$(\exists (rp_m, dr_n) \in RPDRA)$
$[((d_j, op_k), dr_n) \in PDRA \wedge$
$(s_i, rp_m.r) \in SR \wedge$
$rp_m.ER \subseteq \{er \in ER \mid (\exists EC'_l \subseteq EC_l) \ [(EC'_l, er) \in EA]\}]$

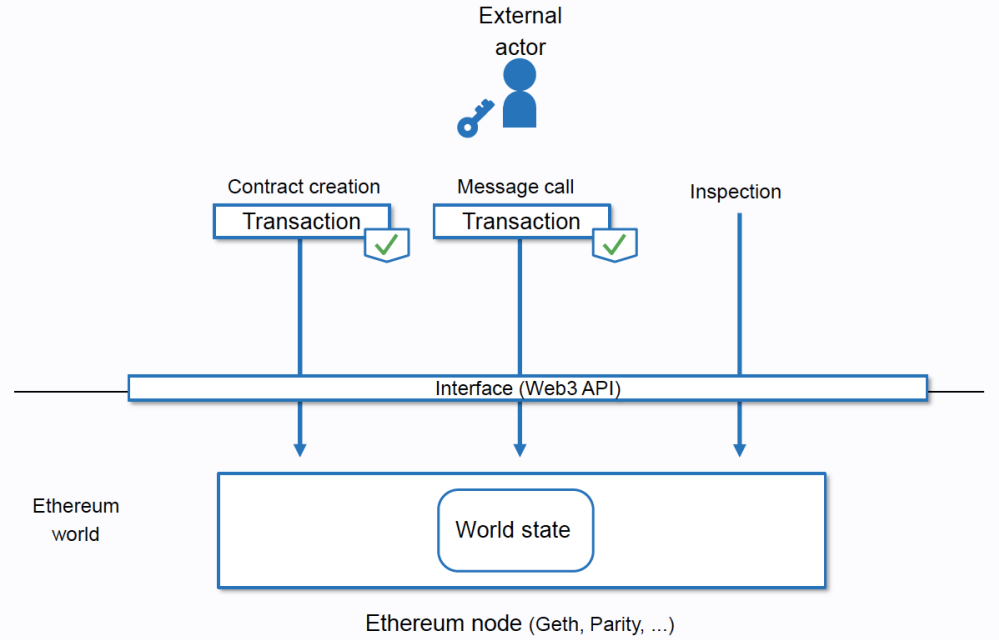

Figure 2.3: Access to Ethereum Network via Interface to a Node, taken from [2]

be activated. So, the user would be granted with the permissions available to the current device roles which are assigned that role pairs for the duration of that session. EGRBAC is an operational model of our choice upon which we would build our administrative model, in that its provided granularity along with context-awareness make it a suitable choice for access control in smart home environments. However, this model is limited to govern only user-to-device accesses and leaves device-to-device access control for future investigation. Correspondingly, the proposed administration model in this dissertation would also inherit the same constraint.

2.2.2 Blockchain-Based Access Control in IoT

Blockchain Preliminaries

Blockchian is well known as an underlying technology for Bitcoin which has been introduced in Satoshi Nakamoto's original whitepaper in 2008 [134]. While there is no specific word of blockchain in the paper, it describes the technology as a series of data *blocks* which are cryptographically *chained* together. Nowadays Blockchain is mostly known as a distributed, decentralized ledger in which blocks of transactions are linked together cryptographically by maintaining a

peer to peer network. As there is no single point of trust/failure, the participating peers usually have to solve a complicated problem to reach consensus with each other and vote to add a new block to the chain in a process which is called *mining* in technical vernacular and the participating nodes are known as *miners*. Different voting systems have been developed which are usually referred to as *consensus algorithms*. In order to avoid malicious actors, there are different incentives provided to guarantee truthfulness, such as awarding voting with the majority and punishing voting against the majority. Finally, this game-theoretical equilibrium would end up adding a block to the chain which is authentic. Therefore, without the presence of any central authorities, untrusted parties can share a consensus view.

Although Bitcoin was the first application of blockchain, different companies started to develop beyond Bitcoin. There are different classifications of blockchain technology. The basic classification would be *public* and *private* blockchains, which are distinguished based on who is authorized to take part in the network. In public blockchain anyone can join and participate in the blockchain, i.e. see/confirm the transactions and mining blocks. Private blockchains, on the other hand, need to be approved in order to join the blockchain by the specific person or based on a set of rules. Once a new participant joins, its role would indicate what it can do in maintaining the blockchain. Another basic division that is made between different types of blockchain is that of permissioned and permissionless blockchain. Permissionless blockchains need no permission to participate in the network, however in permissioned blockchains a party needs to be somehow privileged to play a role. These definitions are far from clear as sometimes public and permissionless blockchains are considered the same, as private and permissioned blockchain.

In this dissertation, we opted for Ethereum blockchain as our enforcement tool of our proposed administrative model for smart home IoT in Chapter 4. Ethereum is a decentralized and public blockchain which is essentially permissionless which is mostly actively used blockchain based on Bloomberg [1]. Microsoft adopted Ethereum as the core of its Blockchain-as-a-Service (BaaS)

[1] https://www.bloomberg.com/news/articles/2020-11-23/ethereum-races-clock-to-collect-enough-coins-for-huge-upgrade

on the Azure cloud computing environment [2], however Ethereum could be run in any distributed environment. As it is permissionless, anyone would be able to spin up a node, join the Ethereum network and broadcast transactions or mine blocks. Decentralized nodes constitute the Ethereum P2P network.

Unlike permissioned blockchain, there is not a central authority available on the network level in Ethereum. However, it is possible to define rules on what each person would be permitted to do. Ethereum is featuring smart contract functionality which is used to define rules and penalties and enforces those obligations. Ethereum smart contracts are autonomous and stateful scripts which would be stored in the Ethereum blockchain and are utilized to define the rules. In Ethereum, smart contracts are treated as autonomous scripts or stateful decentralized applications that are stored in the Ethereum blockchain for later execution. Ethereum Virtual Machine (EVM) is the runtime environment of smart contracts which has been formally defined in the Ethereum Yellow Paper [204].

External actors access Ethereum through Ethereum Nodes, the overall view of access to an Ethereum network's node has been depicted in Figure 2.3. World state is a mapping between address and account state, i.e. each Ethereum account is considered as an object in Ethereum's world state. There are two types of accounts in Ethereum: Externally Owned Account (EOA) which could be controlled by anyone in possession of a private key and could be created free of charge. The other type of Ethereum account is the Contract Accounts (CA) which uses smart contract (EVM) code, creation of which has a cost because it uses the network's storage.

Any external account is able to create a single cryptographically signed instruction, which is called a transaction. A transaction could be used to create a new contract (CA) or it could trigger a message [204]. Any transaction needs to be approved as valid through mining, which costs a fee. There is a universally agreed upon fee for any specific type of instruction (account creation, executing instructions on EVM, access to account storage, etc.) which is expressed in *gas*. Gas prices would be paid in Ethereum currency, namely *Ether (ETH)* and expressed in terms of *Wei*,

[2]https://www.coindesk.com/microsoft-launching-new-ethereum-blockchain-product

$1\ Wei = 10^{-18} * ETH$. Transactions would be batched in blocks, which would be mined and successfully added to the blockchain. Each successfully mined block would be broadcasted to all the network participants. Current protocol of mining in the Ethereum blockchain is a Proof of Work (PoW).

Blockchain and IoT Access Control

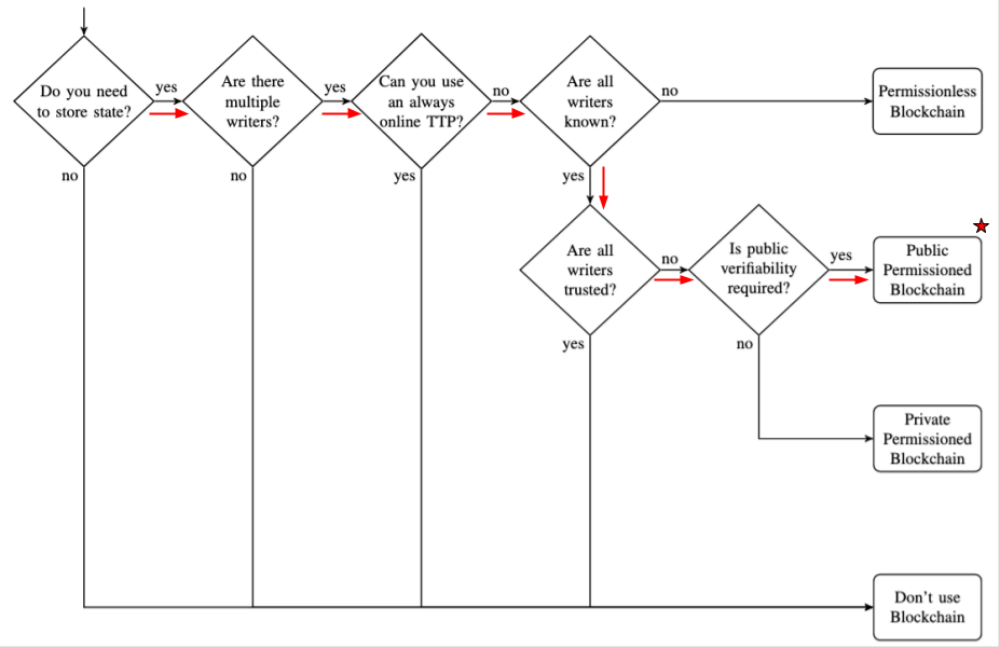

Figure 2.4: Whether a Blockchain is the Appropriate Technical Solution for Your Problem, taken from [205]

Numerous access control frameworks have been proposed based on blockchain, given the intense hype around this technology. Ethereum [8] is the first blockchain platform to present the smart contracts [45] and provides a built-in Turing-complete programming language, i.e. Solidity, which makes it possible to create arbitrary state-transition functions on blockchain to encode intended logic. Other blockchain platforms like Hyperledger Fabric [3], Ripple [4], bitcoin [5] and EOS [6] have later provided smart contract capability to their chains. However, being the first platform to

[3]https://www.hyperledger.org/use/fabric
[4]https://www.ripple.com/
[5]https://bitcoin.org/en/
[6]https://eos.io/

provide smart contracts along with the maturest code-base and user-base, Ethereum has been used in many access control frameworks which have been proposed to provide distributed IoT access control.

Blockchain is utilized in many research works to implement operational access control. Some authors considered traditional access control models, including RBAC and ABAC as the most prominent ones, to have some drawbacks which makes them inappropriate in many IoT use cases, as listed below [207]:

- Scalability: management burden for traditional models, especially RBAC and ACL, makes them hard to scale as the number of IoT devices rapidly grows. The complexity of management grows in ABAC when large-scale IoT environments are needed, thus attribute management brings added intricacy.

- Heterogeneity: Both RBAC and ABAC paradigms are inflexible to manage delegation and transitivity for intra-domain communication. This problem would get worse in IoT environments as there are many different vendors and IoT platforms.

- Spontaneity: traditional access control models, including RBAC and ABAC, are designed with a presumption of a long-lived pattern to exist. However, IoT environment interactions are usually spontaneous, short-lived and highly dynamic.

- Interoperability: As IoT devices are resource constrained and usually communicate with each other through lossy networks, a lightweight access control would be an outstanding requirement.

On the other hand, the blockchain characteristic of being naturally distributed removes the single point of failure and other problems associated with central management. Blockchain immutability eliminates the concern of privacy leakage by untrusted third parties and using consensus protocols ensures to have only valid transactions recorded on the chain. Moreover, immutability makes auditable access control possible. Furthermore, the idea of smart contracts to express ar-

rangements between parties into tamper-proof, self-executing computer codes found extensive use cases in automated access control [129].

Despite all the benefits which could be taken from employing blockchain in access control solutions, there are also some researchers who have a more skeptical view on the blockchain phenomenon and the excessive hype around it. Authors in [205] argue the use of blockchain to be sensible only when there are multiple mutually mistrusted parties who wants to make changes in a system's state and they do not want to rely on a third part for this purpose. Based on this argument, authors proposed a flow chart as shown in Figure 4.4 which tries to determine the necessity/type of the blockchain to be used. Red arrows in the Figure 4.4, indicate our position with blockchain requirement and utilization, as we propose to use Ethereum blockchain for administrative access control in smart home IoT.

BlendCAC [207] encodes access rights in capability tokens which are deployed as smart contracts along with another type of smart contract which represents the delegation. The proposed capability-based model is at operational level in which each transaction required to spend almost $1.02 to be completed given the gas price in the public Ethereum in 2018, which is strongly prohibitive to be used by a normal user of a smart home even if the Ether price did not spike. Authors in [135] tried to fix some of the issues of the BlendCAC model by proposing a fine-grained access control model, however no cost or performance metric has been discussed. Another blockchain-based, capability-based approach could be found in [89] which is a decentralized user-centric approach based on the publish-subscribe model.

An attribute-based access control for IoT environments has been proposed in [87]. Authors tested the proposed frameworks not on a local network, but on one of Ethereum test networks called Rinkeby. Rinkeby uses Proof of Authority (PoA) as its consensus mechanism, which is faster than current Ethereum consensus mechanism, i.e., Proof of Work (PoW). So, presented results in [87] could be considered as a lower bound. Yet, those costs are still prohibitive to be applied in a smart home for operational access control. Another approach [126] has been codified and tested on the Ropsten (PoW) testnet, which uses the same consensus algorithm as Ethereum

mainnet (main network). However, the estimated space and gas requirements in the paper proves the proposed approach to be nonviable in a smart home environment.

Role-Based access control has been also used to design blockchain-based approaches for mediating access in IoT environments [55, 159]. Cruz et. al. [55] proposed an RBAC-based platform along with a challenge-response protocol to facilitate inter-organizational access control. Since their RBAC-based smart contract only encodes add/remove a user/endorser and change the contract's status, which we argue as administrative-type of tasks, the evaluation results show this platform to be practical in terms of cost, however authors have not provided any time evaluation.

There are some blockchain based access control frameworks lacking the basis of a formally defined access control models, such as the trust-based layered framework proposed by Dorri. et al. [65] or the RDF-based architecture for IoT access control in smart buildings [41]. ControlChain [155] is another blockchain based architecture for IoT authorization which does not rely on a specific access control paradigm, instead authors included an encoder using which different access control models could be transformed to their architecture authorizations. Furthermore, some blockchain-based access control frameworks are built upon other blockchain platforms, such as bitcoin [125, 146], Hyperledger Fabric [104, 123] and EOS [159].

2.3 Device-to-Device Communication in IoT Environments

Our proposed model in Chapter 5 aims to provide an access control model for device-to-device interoperability in an IoT smart home which contains heterogeneous devices. Although there is no access control model specification proposed for this purpose, we briefly discuss some of the academic and industrial efforts to facilitate device-to-device communications in different IoT application domains.

Seamless interoperability among IoT devices, end-to-end device-to-device communication, is imperative for incipient evolution of the IoT ecosystem. There are situations in which co-located devices that want to interoperate use heterogeneous communication technologies, which makes it challenging [33, 140]. There are extensive efforts going on to provide direct intelligent communi-

cations among IoT devices. OneM2M [197] is a community aimed at providing common machine-to-machine services by developing technical specifications. IEEE P2413 [101] defines reference architectures for multiple IoT domains, smart city, transportation, etc. The 3rd-generation partnership project (3GPP) to provide cellular IoT via proposing innovations on Extended Coverage GSM Internet of Things (EC-GSM-IoT), LTE for Machine-Type Communications (LTE-M) and Narrowband Internet of Things (NB-IoT) [121]. Some researchers proposed authentication and authorization mechanisms based on 3GPP [32,179] at networking layer and focused on user's authentication. Similarly, there are a number of approaches which rely on semantic models based on OSGi framework [21] to provide network-level interoperability in home environments [42,107,152]. However, none of them give a clear account on authorization and access control modeling.

Furthermore, there are multiple standardization efforts going on to define architectural standards to facilitate IoT systems interoperability. BIG IoT [162] was a joint project among 13 European participants, trying to address the fragmentation of IoT application/services by designing a unified Web API for smart city applications, called the BIG IoT API which is a software library. The project results contained in [30,43,44,175]. Agile [70] is anther project which tries to provide device-level interoperability through building an adaptive and modular gateway for heterogeneous IoT devices [73,119]. SymbIoTe (symbiosis of smart objects across IoT environments) [164] provides an abstraction layer to enable resource sharing and discovery on top of various IoT platforms via a unified control view for rapid cross-platform applications [83, 189, 196]. Authors in [178] proposed a distributed authorization system, which relies on SymbIoTe ABAC-based authorization and allows platforms to define access policies at resource level. However, all nodes must have the same access control data and agree on the same result, as proposed work cannot provide trustworthiness. Nevertheless, no access control model/specification has been provided.

Vicinity [71] is another project aimed at connecting different smart objects into a social network, a.k.a virtual neighborhood, to build a device- and standard-agnostic platform for IoT infrastructure [52, 85], with use case in energy, building, e-health and mobility application domains. BIoTope [163] tries to build a set of necessary standardized open APIs to provide system inter-

connections using contextual information from heterogeneous platforms with large-scale pilots implemented in smart cities. OpenIoT is an open source IoT platform proposed to facilitate data collection from a variety of sensors. It provides semantic interoperability of heterogeneous platforms by providing a standard-based model via applying a SSN-ontology [180, 188]. Nonetheless, using OpenIoT requires all resources to be integrated with the same information model, namely SSN ontology. All the aforementioned efforts have a common goal , which is to create standards for heterogeneous IoT device interoperability. While having commercial and real-world partners, however, there is no de-facto standard and there would be none in foreseeable future [127].

INTERIoT [77] is an approach which is not aimed for providing a reference nor a standard model for IoT communication. Rather, its goal is seamless cooperation/integration of heterogeneous IoT platforms based on a layered approach. In a device-to-device paradigm, all devices could communicate seamlessly with each other without intermediaries, through their local area network or over the Internet. There are a few research works in which authors tried to embed authorization logic in the smart objects by using capability tokens [35, 96], however these approaches are more focused access enforcement, and no access control model specification is proposed. Moreover, proposed model has low awareness of the contextual information for making access decisions which is very important to be considered in dynamic IoT environments. However, the heterogeneity in all levels of IoT technology, including device, networking, middleware, application and data/semantics of IoT scenarios, makes heterogeneous IoT devices cumbersome [77]. Enabling technologies for next generation smart systems which includes device-to-device communications and its challenges have been discussed in [182].

Another group of research works considered heterogeneity of IoT devices, even so their focus is on user-to-device interactions over heterogeneous IoT platforms [143]. A capability-based access delegation approach for device-to-device interaction has been proposed in [28]. However token validation is not addressed and it contains no access control model to support device-to-device authorization. It is noteworthy that many IoT devices are known as *black boxes*, as their state and composition, including the identity of external services with which they communicate, are not

visible. In many cases there is no access to IoT devices software or configuration [130]. Therefore, IoT devices call for specific ways of access and management, which would probably be different than conventional ways for common IT devices. In a device-to-device paradigm one IoT device requests access to another device and there should be an appropriate access control to govern that.

Chapter 3: SAFETY AND CONSISTENCY OF SUBJECT ATTRIBUTES IN DISTRIBUTED ABAC ENVIRONMENTS: A SMART HOME USE CASE

Attribute-based access control (ABAC) systems typically enforce pre-authorization, whereby an access decision is made once prior to granting or denying access. In this chapter, we address the safety and consistency problem in distributed attribute-based access control (ABAC) systems. Our focus is on the consistency of subject's attributes. First, we propose five consistency levels by enforcing restrictions on timeliness of revocation checks of attribute certificates. Second, we present a completely new perspective by considering a refresh scenario, instead of revocation. We formally characterize three increasingly strong levels of consistency to restrict the exposure of the decision point to stale attribute values. Finally, we investigate the safety and consistency problem in the context of mutable attributes in the sense of the UCON_{ABC} model [150]. Mutability adds further complication to establish consistency requirements and specifications, as it requires synchronization mechanisms in place to update attribute values. We identify two categories of use cases of practical benefit in the context of ABAC, which turn out to be amenable to quota-based solutions. We develop a formal characterization of required consistency using refresh in this context. We also observe that revocation is inappropriate to be used in the context of mutable attributes, which has been the traditional approach for checking the attributes' freshness for immutable attributes. Results of above-mentioned research have been published in following three peer-reviewed papers:

1. Mehrnoosh Shakarami, and Ravi Sandhu. "Safety and Consistency of Subject Attributes for Attribute-Based Pre-Authorization Systems", In Proceedings of National Cyber Summit (NCS), pp. 248-263. Springer, Cham, 2019.

2. Mehrnoosh Shakarami, and Ravi Sandhu. "Refresh instead of revoke enhances safety and availability: A formal analysis." In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 301-313. Springer, Cham, 2019.

3. Mehrnoosh Shakarami, and Ravi Sandhu. "Safety and Consistency of Mutable Attributes Using Quotas: A Formal Analysis." In 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), pp. 1-9. IEEE, 2019.

3.1 Motivation

In attribute-based access control (ABAC) access decisions are made on the basis of attribute values of subjects, objects and environment with respect to a policy. Attributes and policy are susceptible to change. Ideally the decision point should know their real-time values, which is not practically feasible. Even if attributes and policy are queried from appropriate authorities immediately prior to every decision, there is irreducible network latency. Realistically, some values will be cached due to performance, cost, and failures. Consequently, some access decisions may be incorrect. We call this the safety and consistency problem. We investigate this problem with focus on subject attributes. We assume that the policy and object/environment attributes are known in real-time at the decision point. This is reasonable since the decision and enforcement points are typically co-located with the object's custodian who maintains these values. This reduces the problem to safety and consistency of subject attributes.

Moreover, we note recognize the concept of refreshing attribute values rather than simply checking the revocation status, as in traditional approaches. Refresh replaces an older value with a newer one, while revoke simply invalidates the old value. As we will show, this enhanced possibility of getting a new value rather than an invalid response enhances safety and availability. We also define the concept of being satisfactory for an attribute value with respect to a policy, which is first introduced in our work to the best of our knowledge. Relying on the history of satisfactory attribute values, we introduce additional flexibility to grant access to authorized users. Lastly, we investigate safety and consistency in the context of mutable subject attributes which introduces additional complexity to the problem. We consider attributes mutable if their changes are the consequence of access utilization by subjects. In particular, there might be multiple concurrent sessions manipu-

lating the same mutable attribute. Therefore, in addition to exposure of the decision point to stale attribute values, safety and consistency can be compromised due to concurrent utilization of the same attribute. While the general consistency problem has vast literature in the distributed systems domain, practical solutions are typically dependent on the specific application domain. We identify two categories of use cases of practical benefit in the context of ABAC, which turn out to be amenable to quota-based solutions. We provide a formal analysis of the resulting solutions. We provide smart home use cases in each section to showcase how safety and consistency problems could be handled utilizing proposed solutions.

3.2 Safety and Consistency of Subject Attributes for Attribute-Based Pre-Authorization Systems

Assuming an ABAC model is in place, we focused on a pre-authorization model in which our goal is to provide the decision point with the most recent status of subjects' attributes. For convenience we understand the term attributes to mean attribute values. These attributes are obtained as credentials (a.k.a certificates) issued by an Attribute Authority (AA). A credential which must be coupled to a specific subject, which is typically achieved by embedding the subject's identity in the credential. The identity of the subject must be authenticated before the credential is coupled with that subject. The details of these processes can be complex and susceptible to security vulnerabilities and flaws. All the same there are multiple well-known standards such as X.509 [137], SAML [142] and OAuth [102] in this arena. We assume that suitable mechanisms exist to bind credentials to subjects without requiring any specific technique for this purpose. Regardless of the way through which the attributes are presented, we assume proposed attributes to be authentic and tied to the subject. Credentials may be signed or unsigned depending on how they are acquired. Common to all definitions, a credential specifies the value of a single attribute for the subject.

Each credential has a start time and end time, which establish the overall lifetime for validity of the attribute value given in the credential. The credential may be revoked by the AA during this putative lifetime. Thus the relying party (the decision point) must make one or more revocation

Table 3.1: Table of Symbols

Symbol	Meaning	Symbol	Meaning
c_i	i^{th}credential	t_{req}	request time
$t^i_{r,k}$	time of k^{th} revocation check for c_i	t_d	decision time
$t^i_{r,max}$	last time of revocation status check for c_i	t_e	enforcement time
$t^i_{invalid}$	first time c_i has been found to be revoked	t^i_{start}	start time of c_i
t^i_{revoc}	actual revocation time for c_i (if any)	t^i_{end}	end time of c_i

checks with the AA to gain additional assurance of the credential's validity [1]. The consistency problem arises when multiple credentials of a subject are required to make an access decision. If the lifetimes and revocation check times of required credentials do not align properly it is possible to make incorrect access decisions, both in allowing access that should be disallowed and vice versa.

3.2.1 Problem Statement and System Assumptions

We require every credential to have a determined lifetime interval which has been specified by its *start time* and *end time*. For short-lived credentials this interval is small, say minutes, seconds or even less, while for long-lived credentials this interval could span days, months or years. In either case we recognize the possibility that the credential may get revoked during its lifetime, although this is especially germane for long-lived credentials. We forbid use of a credential outside its lifetime. The revocation status of a credential may be checked as appropriate by the decision point, and a credential that is known to be revoked cannot be unrevoked. We also assume that attribute values do not change as a result of credential usage, so that attributes are immutable in the sense of [150].

Following LW we refer to the set of a subject's credentials used to make an access decision as the *view* of the decision point (V_{DP}). The appropriate view depends upon the policy being evaluated. In general, the view might change during evaluation. Consider a policy P which is a

[1]Credential lifetimes can range widely from months to seconds. For very short-lived credentials revocation checks may not be useful. For simplicity we consider that for a short lived credential there is an implicit and successful revocation check at its start. Thus we can uniformly assume there is at least one revocation check by the relying party for each credential that it uses in making an access decision. For long-lived credentials there should be at least one revocation check after start time.

disjunction of two predicates A and B, i.e., $P = A \vee B$. The decision point may choose only credentials included in the A predicate as relevant to P in order to perform the first step of evaluation; if A fails, B will replace it in the view and the set of relevant credentials to P will change as a result.

Definition 1. The set of attributes included in the view of decision point related to the policy P at time t is called the set of relevant credentials and denoted by $V_{DP}^{P,t}$.

Since required credentials for evaluating a policy would be collected incrementally and lifetimes of different credentials might not be the same, there is no guarantee that previously collected credentials are still valid while the latter ones are acquired, which might cause the safety and consistency problem. Following example illustrates the inconsistency problem.

Example. Alice is a portal manager in the sales department of a company. If she wants to communicate with clients through the portlet website, the decision point needs credentials attesting her sales group membership and user role. If she wants to utilize higher levels of access, for example editing and approving contracts with clients, both user and manager role credentials are required. Suppose Alice has the user role from January 1st (start time) until March 1st (end time). Her sales group member certificate is valid from January 25th till February 24th and she is given the manager role from Feb 10th (start time), which is valid until March 9th. Suppose the decision point acquired and validated the user role and sales group certificates most recently on January 25th and Feb 8th respectively. It also collected a manager certificate on Feb 10th which was verified to be valid (via revocation check) on the same day. So, the decision point would honor the manager's access to Alice if she requests an access afterward. Due to a reorganization in the company, Alice may no longer be a manager after Feb 17th. Also, suppose Alice's user role certificate has been prematurely revoked on Feb 9th. But if the decision point still relies on the previous revocation checks, Alice would be able to exercise manager's rights after revocation of her relevant credentials, which results in an access violation.

Violation by relying on outdated validation information is a common problem in access control enforcement. While this example illustrates inconsistency with long-term cached credentials, sim-

ilar problems can arise even if all needed credentials are accumulated over a short period of time. Table 3.1 defines a set of self-explanatory symbols to refer to important time stamps used in this section. There are some common assumptions which have been made in both LW and our work as follows.

1. Once a credential is revoked, it cannot be unrevoked. However, a new credential can be issued for the same attribute for that subject.

2. There is a single instantaneous decision time t_d. The access decision may be re-evaluated subsequently with, say, different credentials but this latter evaluation is treated as a separate and distinct decision.

3. V_{DP}^{P,t_d} is the only view of interest, in which P is the policy which should be satisfied to grant the access at decision time t_d and DP stands for the decision point.

We always use the latest revocation check results for making access decisions. So, if we have max_i revocation checks for c_i, the $t^i_{r,max}$ indicates the latest revocation check ($max_i{}^{\text{th}}$) of c_i and it would be utilized in decision making.

3.2.2 Consistency Levels

In this section, we develop five consistency levels relative to the view of the decision point ordered as shown in Fig. 3.1-(a). We say that a credential is in its validity interval or is *valid*, provided that the current time is not before the credential's start time, nor after the credential's end time and the credential is not known to be revoked at any time before the current time. A revocation check is never done after the end time since the credential has already expired. Once it is revoked a credential cannot become valid again. It follows that if validity of a particular certificate is confirmed via revocation check at time t after its start time, the credential has been valid for all times between its start time and t. Let C be the set of all credentials in the system, and T the set of all possible time stamps. We formalize the notion of a credential's validity status at time t by defining the following three-valued function. We call a credential $Invalid$, if the following function

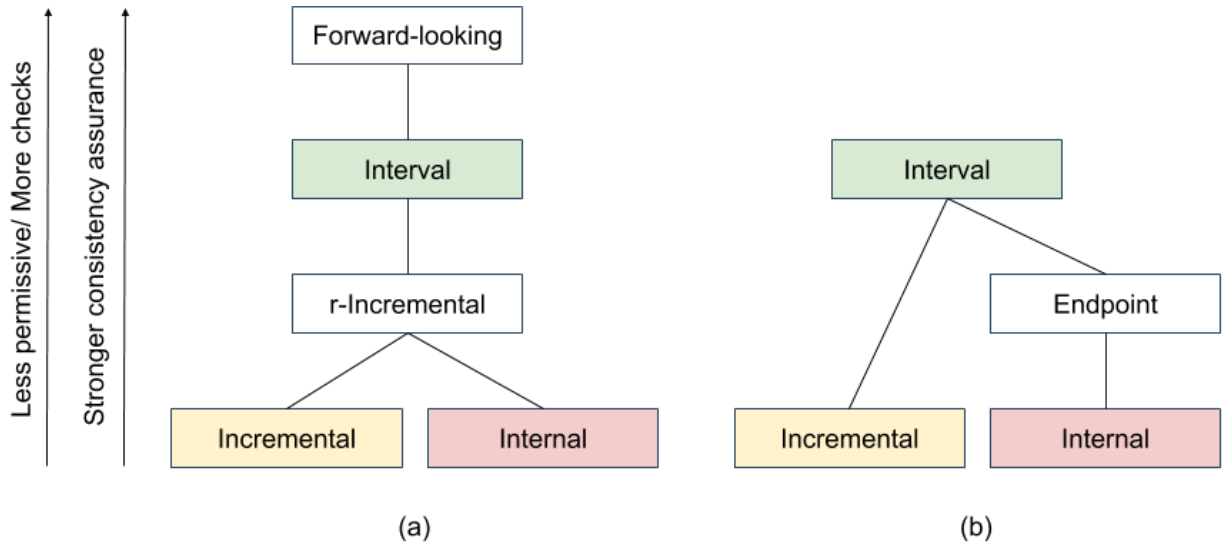

Figure 3.1: (a) Our consistency levels (b) LW consistency levels. Equivalence is color coded.

returns $False$.

$$Valid : C \times T \rightarrow \{True, False, Unknown\}$$

$$Valid(c_i, t) = \begin{cases} True \iff & Valid(c_i, t^i_{r,k}) \wedge (t^i_{start} \leq t \leq t^i_{r,k}) \\ Unknown \iff & Valid(c_i, t^i_{r,max}) \wedge (t^i_{r,max} < t \leq t^i_{end}) \\ False \iff & (\neg Valid(c_i, t^i_{r,max}) \wedge (t \geq t^i_{r,max})) \\ & \vee (t \notin [t^i_{start}, t^i_{end}]) \end{cases} \tag{3.1}$$

In the rest of this section we propose five consistency levels. For each consistency level, we provide a formal specification along with the properties which are guaranteed if we apply the proposed specification.

Incremental Consistency

This level requires each relevant credential to be found valid by a revocation check before the decision time. In the above-mentioned example in Section 3.2.1, suppose Alice wants to access the portal on Feb 25th, so user and sales group certificates should have been checked. Although

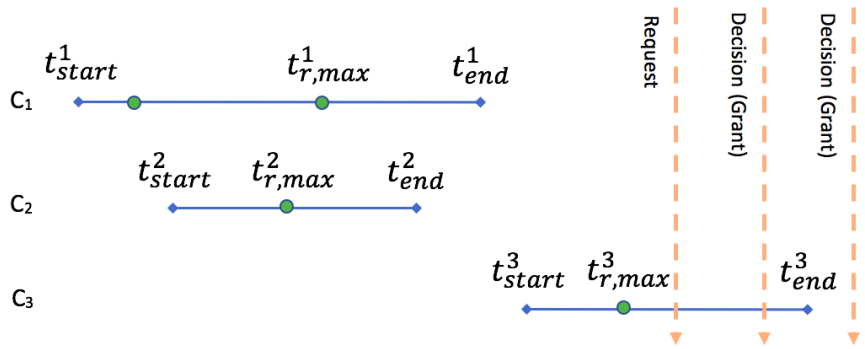

Figure 3.2: Incremental Consistency with Unrestricted Decision Time

one of her relevant credentials (sales group) expired one day ago, the system would let her in. This access violation happens because at this level we may use a credential for an access decision after its t^i_{end} time. As shown in Fig. 3.2, two credentials C_1 and C_2 have been used after their corresponding end times.

Specification. Every credential in the view of decision point is valid at its latest revocation check which has been done before the decision time.

$$(\forall c_i \in V_{DP}^{P,t_d})\ [(t^i_{start} \leq t^i_{r,max} < t^i_{end}) \wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{r,max} < t_d) \wedge Valid(c_i, t^i_{r,max})] \tag{3.2}$$

Property1. For every relevant credential, there is at least one point in time before the decision time, at which that credential has been found (via revocation check) to be valid.

$$(\forall c_i \in V_{DP}^{P,t_d})(\exists t_i)\ [(t^i_{start} \leq t_i < t^i_{end}) \wedge (t_i < t_d) \wedge Valid(c_i, t_i)]$$

Proof. Without loss of generality, we can assume $t_i = t^i_{r,max}$. Moreover, we know that $\max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{r,max} < t_d \implies t_i < t_d$.

Internal Consistency

In order to enforce lifetime overlap for all relevant credentials, internal consistency requires every relevant credential to be started before the end point of any other relevant credential. Furthermore, if a credential is revoked, this revocation should happen after all credentials have started. As shown

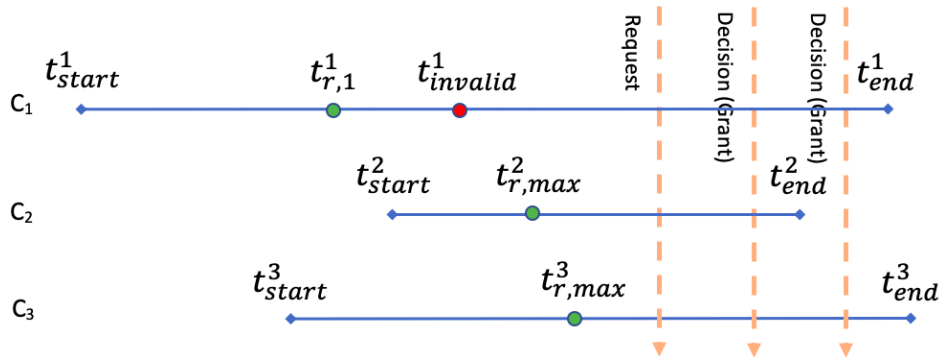

Figure 3.3: Internal Consistency

in Fig. 3.3, it is possible to deliberately utilize an already revoked credential at this level. Moreover, it is still possible to use a credential beyond its end time, as in incremental consistency. In the case of example in Section 3.2.1, Alice would be granted access to the portlet even if we know her user role credential has been revoked on Feb 9. The formal specification is as follows.

Specification. Every credential in the view of decision point has to be started before the minimum endpoint of all credentials and has to be valid at some point before the decision time. The minimum known revocation of any relevant credential occurs after all credentials have been started.

$$(\forall c_i \in V_{DP}^{P,t_d})(\exists t^i_{r,k})\ [(t^i_{start} \leq t^i_{r,k} < t^i_{end}) \wedge Valid(c_i, t^i_{r,k}) \wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{r,max} < t_d) \\ \wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{invalid}) \wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end})] \tag{3.3}$$

Property1. There is at least one point in time at which all relevant credentials are in their $[t_{start}, t_{end})$ time intervals and are not known to be *Invalid*.

$$(\exists t')(\forall c_i \in V_{DP}^{P,t_d})\ [(t^i_{start} \leq t' < t^i_{end}) \wedge (Valid(c_i, t') \neq False)]$$

Proof. The last condition in Equation 3.3 provides overlapping of lifetimes of all relevant credentials. Also, there is at least one revocation check for every credential at which it has been found to be valid. So, there is at least one point, namely t', in intersection of lifetime intervals of all credentials at which every credential is either checked and found to be valid before t' (its validation status is unknown at t') or it has not been checked yet (so it is valid at t'). If

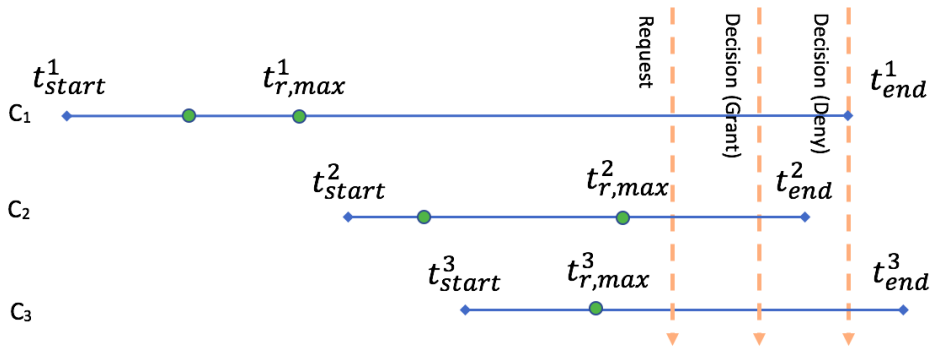

Figure 3.4: Incremental Consistency with Restricted Decision Time

there is any credential which has been found to be revoked, t' should be picked from the interval: $t' \in [\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{invalid})$.

Property2. There is no subset relationship between incremental and internal consistency levels.

Proof. It is possible to have an incrementally consistent view in which there is no overlap between lifetime intervals of all relevant credentials. So, it would not be internally consistent. On the other hand, there may be an internally consistent view at which we recognize a credential at its latest revocation check to be prematurely revoked, so thereby not incrementally consistent.

Incremental Consistency with Restricted Decision Time (Restricted-Incremental or r-Incremental)

In this level, we restrict the decision time to happen necessarily when all relevant credentials are in their lifetimes, say $[t^i_{start}, t^i_{end}]$ (Figure 3.4). As opposed to previous levels, in this level if any of the relevant credentials has expired the access request would be denied. In the case of Section 3.2.1, if Alice tries to exercise her rights on Feb 25th (after her credential expiration) the decision point would deny her access. In Figure 3.4, the second decision time would result in Deny, comparing with the similar situation in Figure 3.2 and Figure 3.3 where both access requests resulted in Grant. Specification and guaranteed properties are given below.

Specification. Every relevant credential has to be found valid at the latest revocation check which, by assumption, happens before the decision time. Moreover, it is essential that the decision time

happens before any of the relevant credentials end time.

$$(\forall c_i \in V_{DP}^{P,t_d})\ [(t^i_{start} \leq t^i_{r,max} < t_d < t^i_{end}) \wedge \mathit{Valid}(c_i, t^i_{r,max})] \tag{3.4}$$

Property1. There is at least one point in time at which all the relevant credentials are in their $[t_{start}, t_{end})$ time intervals and are not known to be $\mathit{Invalid}$.

$$(\exists t')(\forall c_i \in V_{DP}^{P,t_d})\ [(t^i_{start} \leq t' < t^i_{end}) \wedge (\mathit{Valid}(c_i, t') \neq \mathit{False})]$$

Proof. Based on Eq. 3.4, $(\forall c_i \in V_{DP}^{P,t_d})\ [t^i_{start} \leq t_d < t^i_{end}]$. So, $\max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{start} \leq t_d < \min_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{end}$. By taking $t' = t_d$, the proof for the first part is trivial. For the second part, we know that the latest time we checked c_i's revocation status is $t^i_{r,max}$, at which we found it to valid (otherwise the access would be denied). But, we do not know about the real status of the credential after the last revocation check and the Valid function would return $\mathit{Unknown}$ at these later times.

Property2. Any incrementally consistent view with restricted decision time has the following property: $\bigcap_{\forall c_i \in V_{DP}^{P,t_d}} [t^i_{start}, t^i_{end}) \neq \varnothing$

Proof. Following previous proof, there is at least one point (t_d) that lies in the $[\max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{start}, \min_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{end})]$ interval. So, this interval is not empty.

Property3. Any r-incremental consistent view is incremental and internal consistent as well.

Proof. All three specifications have it in common that every relevant credential has to be found valid at its revocation check. The first part of the incremental consistency with restricted decision time is:

$$(\forall c_i \in V_{DP}^{P,t_d})\ [t^i_{start} \leq t^i_{r,max} < t_d < t^i_{end} \implies (t^i_{start} \leq t^i_{r,max} < t^i_{end})$$
$$\wedge\ (\max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{r,max} < t_d) \wedge\ \exists t_d \in \bigcap_{\forall c_j \in V_{DP}^{P,t_d}} [t^i_{start}, t^i_{end}]$$

Therefore, r-incremental is a constrained version (subset) of incremental level. Moreover, since we use the latest valid status, we are not aware of any revocation and $t^i_{invalid} = \mathit{Null}$. So, all

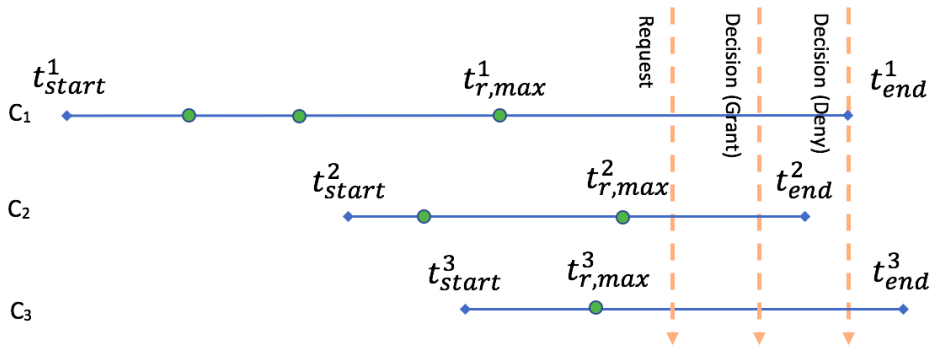

Figure 3.5: Interval Consistency

properties of the internal level are also satisfied.

Property4. It is not necessarily the case that any incrementally/internally consistent view is r-incremental as well.

Proof. In both incremental and internal levels, the decision time may be after some of the relevant credentials' endpoints, which means that we may have: $t_d > \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end}$, which contradicts r-incremental specification.

Interval Consistency

In case of the example in Section 3.2.1, Alice's user role has been revoked on Feb 9th. If she tries to communicate at manager level with clients at that date and system still relies on the latest revocation check which happened before actual revocation she would be let in, while there is no guarantee that the credential is still valid. We know her user role has been revoked even before the manager certificate starts. Interval level enforces latest revocation checks to happen in $[t^i_{start}, t^i_{end}]$ for all relevant credentials. So, it could be guaranteed that not only every credential is valid at some time, but also all credentials were simultaneously valid. The specification and properties guaranteed by this level are given below.

Specification. Every relevant credential has been found to be valid at the latest revocation check before the decision time. Moreover, the latest revocation check happened after all credentials have

been started and before any of them ends.

$$(\forall c_i \in V_{DP}^{P,t_d})\ [(\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start} \leq t^i_{r,max} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end}) \wedge Valid(c_i, t^i_{r,max})] \tag{3.5}$$

Property1. There is at least one point in time, after all relevant credentials have been started and before any of them ends, prior to decision time, at which all of the relevant credentials are simultaneously valid.

$$(\exists t')(\forall c_i \in V_{DP}^{P,t_d})\ [(\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start} \leq t^i_{r,max} < t' < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end}) \wedge Valid(c_i, t')]$$

Proof. Let $t' = \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{r,max}$. For every relevant credential to the policy, we could guarantee that it has been valid at t'. Note that if any credential has been found to be revoked at t', it cannot be unrevoked at any later time. Therefore, the proof is complete.

Property2. Every interval consistent view is r-incremental.

Proof. It is trivial that: $(t^i_{start} \leq \max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start}) \wedge (t^i_{end} \leq \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end})$. Substituting these equation in interval specification in Eq. 3.5, we can deduce: $(\forall c_i \in V_{DP}^{P,t_d})\ [t^i_{start} \leq t^i_{r,max} < t_d < t^i_{end}]$. So, interval specification satisfies the specifications of r-incremental.

Property3. Not any r-incremental consistent view is necessarily interval consistent.

Proof. Based on Eq. 3.4, latest revocation of a credential might happen before some credentials start time ($t^i_{r,max} < \max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{start}$) or a credential may be validated after some credentials expiration ($\min_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{end} < t^i_{r,max}$), which contradicts with interval consistency specification.

Forward-looking Consistency

In the example in Section 3.2.1, suppose Alice tries to change the clients' contracts on Feb 17th (the set of relevant credentials includes sales group and manager role credentials). All relevant credentials were checked on Feb 10 at which all have been started and none of them expired yet, so the interval consistency timing constraints would be satisfied. However there is an access violation, because the decision point relied on outdated revocation status information (relying on revoked manager certificate). To solve this problem, we take the request time into account in

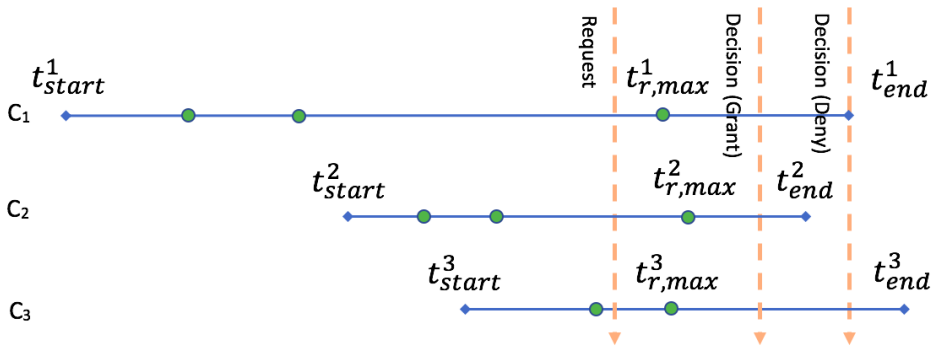

Figure 3.6: Forward-looking Consistency

our strongest level of consistency and impose constraints to ensure all credentials have been valid simultaneously at some point *after the request time* (Fig. 3.6).

Specification. Every relevant credential has to be valid at its latest revocation check time, which happens after the request time and before the decision time.

$$(\forall c_i \in V_{DP}^{P,t_d})[(\max_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{start} \leq t_{req} < t^i_{r,max} < t_d < \min_{\forall c_j \in V_{DP}^{P,t_d}} t^j_{end}) \wedge Valid(c_i, t^i_{r,max})] \tag{3.6}$$

Property1. There is at least one point in time, after the request time and before the decision time, at which all relevant credentials are valid simultaneously based upon their latest revocation checks.

$$(\exists t')(\forall c_i \in V_{DP}^{P,t_d})[(\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start} \leq t_{req} < t' < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end}) \wedge Valid(c_i, t^i_{r,max})]$$

Proof. Suppose $t' = \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{r,max}$. For every relevant credential, we could guarantee that it has been valid at t', because otherwise it cannot be unrevoked at any later time including its latest revocation check. So, the proof is complete.

Property2. Every forward-looking consistent view is interval consistent as well.

Proof. The definition of forward-looking consistency is a restricted version of interval consistency, in which we restrict the latest revocation check to happen necessarily after the request time.

Property3. An interval consistent view is not necessarily a forward-looking consistent view as well.

Proof. In case of interval consistency, it is possible to have a credential c_i with $t^i_{r,max} < t_{req}$, which

contradicts with forward-looking consistency specification.

3.3 Refresh Instead of Revoke Enhances Safety and Availability: A Formal Analysis

As previously mentioned, attribute values in an Attribute-Based Access Control (ABAC) environment are susceptible to change. Ideally the decision point should know real-time values, which is practically impossible due to inherent delays of distributed systems and performance costs. This can lead to granting access when it should be denied (safety violation) or denying access when it should be granted (availability violation). The longer the gap between updates of credentials, the higher the risk of relying on stale attribute values. In this section, we formally characterize three increasingly strong levels of consistency to restrict the exposure of the decision point to stale attribute values. For simplicity, we develop our formalism based on changing subject attribute values. Extension to changing object and environment attribute values is straightforward. Extension to policy changes is more subtle. Policy changes may require additional credentials to come into play. While acquiring these additional credentials the policy may change again. In principle, this could lead to an infinite regress. In practice such an infinite regress is unlikely. Policies composed of multiple sub-policies specified by different authorities also raise issues of policy conflicts [50,124]. A formal treatment of policy changes is beyond our scope.

Our main contribution is to develop a formal framework for safety, availability and consistency problems of ABAC systems, via introducing the refresh scenario instead of the traditional revocation check. As we will show, this enhanced possibility of getting a new value rather than an invalid response enhances safety and availability. We also define the concept of being satisfactory for an attribute value with respect to a policy, which is first introduced in our work to the best of our knowledge. Relying on the history of satisfactory attribute values, we introduce additional flexibility to grant access to authorized users.

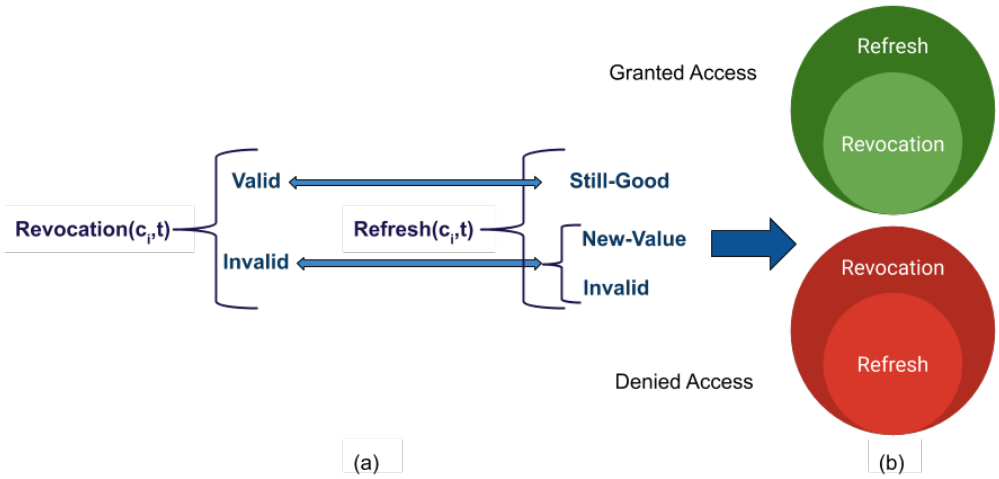

Figure 3.7: (a) Revocation vs. Refresh (b) Comparing Grant vs. Deny

Table 3.2: Summary Table of Symbols

Symbol	Meaning
t_{req}	request time
t_d	decision time
c_i	i^{th}credential
t^i_{revoc}	actual revocation time of c_i (the AA always knows this time)
$t^i_{ref,k}$	time of k-th refresh of c_i
$t^i_{start,k}$	attribute start time of c_i after k-th refresh
$t^i_{end,k}$	attribute expiration time of c_i after k-th refresh
$kmax(t)$	latest refresh of c_i before time t (c_i is determined by context)
$val^i_{kmax(t)}$	the value of c_i after $kmax(t)$-th refresh
$t^i_{ref,kmax(t)}$	time of $kmax(t)$-th refresh of c_i
$t^i_{start,kmax(t)}$	attribute start time of c_i after $kmax(t)$-th refresh
$t^i_{end,kmax(t)}$	attribute expiration time of c_i after $kmax(t)$-th refresh

3.3.1 Problem Statement and System Assumptions

We assume an ABAC authorization system in a distributed multi-authority environment. For a particular access request, there is a single decision point which determines whether or not the access is allowed by the access control policy based on attribute values. For convenience we use the terms attribute and attribute value interchangeably. Subject attributes might change during credential lifetime. A change could be a new value, a new lifetime or a premature revocation. In all cases the decision point needs to be updated about the latest changes of the attribute through either revocation or refresh. In revocation, AA would represent the current status of the credential

as either *Valid* (no change) or *Invalid* (otherwise). However, with refresh AA can indicate the credential's status as *Still-Good*, *New-Value* or *Invalid*. *Still-Good* and *Invalid* correspond to *Valid* or *Invalid* in the revocation scenario. *New-Value* reflects any change in credential's start time, end time or new value. So, *Invalid* status in revocation splits in two possibilities of *Invalid* and *New-Value* in refresh (see Figure 3.7-a). Thereby, refresh can allow more accesses than revoke and deny fewer accesses (see Figure 3.7-b).

$Refresh$ function is defined as follows. T is the set of possible time stamps and C represents the set of all credentials in the system. Table 3.2 defines the symbols used in this definition and throughout the paper.

$$Refresh : C \times T \rightarrow \{Invalid, Still\text{-}Good, New\text{-}Value\} \tag{3.7}$$

$$Refresh(c_i, t) = \begin{cases} Invalid & \iff (t \geq t^i_{end,kmax(t)}) \vee (t \geq t^i_{revoc}) \\ New\text{-}Value & \iff (t^i_{start,kmax(t)} \neq t^i_{start,kmax(t)-1}) \\ & \vee (t^i_{end,kmax(t)} \neq t^i_{end,kmax(t)-1}) \vee (val^i_{kmax(t)} \neq val^i_{kmax(t)-1}) \\ Still\text{-}Good & \iff (t^i_{start,kmax(t)} = t^i_{start,kmax(t)-1}) \\ & \wedge (t^i_{end,kmax(t)} = t^i_{end,kmax(t)-1}) \wedge (val^i_{kmax(t)} = val^i_{kmax(t)-1}) \end{cases}$$

Following example highlights the benefits provided by considering refresh rather than revocation. Although granting illegitimate access is considered as a greater risk in many systems, availability is also important in which a legitimate user should not be denied access.

Example. Authorization policy in a coding company grants read access to a project's code to managers and test engineers and read/write access to developers. Alice was a test engineer. But her role has changed to a developer in the same project. Subsequently she submits a write request to the decision point. In revocation, checking her cached role credential results in *Invalid* response since she is no longer a test engineer. So her request would be denied. In refresh, however, *New-Value* response along with a new credential asserting her new role would be returned and access would be granted, as it should be based on policy.

Claim. If a subject can proceed to utilize a requested access in a revocation scenario, it can proceed in a refresh scenario as well. But there are scenarios in refresh-based systems which let the subject proceed, whereas it would be denied in revocation-based systems.

Proof. If nothing changed about a required credential, revocation and refresh would return *Valid*/*Still-Good* respectively. So, the first part of the claim follows. For the second part, it is possible that a required credential has changed with respect to start/end time or the value. So AA response in revocation scenario will be *Invalid* which prohibits subject's access. However with refresh the response would be *New-Value*, so access would be granted (see Figure 3.7).

System Assumptions

Without loss of generality, we suppose that the policy is stated in Disjunctive Normal Form (DNF), which is the disjunction of different conjuncts. The decision point tries to find the first conjunct which satisfies the desired level of consistency. This conjunct is called the *View* of the decision point at any specific time *t* with respect to the policy *P* which we denote as $V_{DP}^{P,t}$. We assume the decision point can instantaneously check the policy and identify the view.

Definition 2. At any time t, we call the set of subject's attributes included in $V_{DP}^{P,t}$ as the relevant credentials.

We make the following assumptions in this section.

1. Attributes do not change as the result of attribute credentials usage, that is we assume attributes to be immutable in the sense of [150].

2. We will not utilize any expired credentials. If any required credential is beyond its end time, decision point polls AA to get a new credential for the attribute.

3. We do not refresh any credential after it has been found to be *Invalid*.

4. There is one instantaneous decision time (t_d) and one instantaneous request time (t_{req}).

5. V_{DP}^{P,t_d} is the only view of our interest as described above.

6. If refresh returns a *New-Value* result, its start time cannot be prior to its previous start time, i.e., $t^i_{start,k} \geq t^i_{start,k-1}$.

7. AA will not return a credential along with *New-Value* which has not been started yet, so, $t^i_{ref,k} \geq t^i_{start,k}$.

3.3.2 Consistency Levels

In this section, we would first discuss some preliminary concepts as follows.

Satisfactory Values

We define an attribute to be *satisfactory* if and only if its value fulfills the policy conditions. For instance if the policy requires the security level to be at least 3, any security level credential with the value greater than or equal to 3 is considered as *satisfactory*. Obviously the same credential may not be *satisfactory* with respect to another policy. We formally define *satisfactory* with respect to a policy P at the specific time t as follows.

Definition 3. The view at time t has the structure $V^{P,t}_{DP} = \bigwedge_{1 \leq i \leq n} F(i)$ in which $F(i)$ is an atomic expression specifying required conditions for c_i's value. We define Sat as follows to determine satisfactory requirements for c_i's value.

$$Sat^{P,t}_{c_i} = True \iff F(val^i_{kmax(t)}) = True \tag{3.8}$$

Freshness

We rely on the freshness concept in the refresh scenario, compared to validity in the revocation scenario. We formally define freshness via ***Fresh*** function as follows. When ***Fresh*** is used in a boolean expression, we understand $Fresh(c_i, t)$ to be $False$ when its value is $Unknown$.

$$Fresh : C \times T \rightarrow \{True, False, Unknown\}$$

$$Fresh(c_i,t) = \begin{cases} True & \iff (t^i_{start,k} \leq t \leq t^i_{ref,k}) \\ & \wedge\, (Refresh(c_i, t^i_{ref,k}) \neq Invalid) \\ Unknown & \iff (t^i_{ref,k} < t < t^i_{end,k}) \vee (t \geq t^i_{ref,kmax(t)}) \\ False & \iff [(t \geq t^i_{ref,k}) \wedge (Refresh(c_i, t^i_{ref,k} = Invalid))] \\ & \vee [t \geq t^i_{end,kmax(t)}]) \end{cases} \tag{3.9}$$

Following example is used throughout this section. Then, we introduce three levels of consistency taking both old and new values of relevant credentials into account. We provide specifications and consequent properties guaranteed by each level in the rest of this section.

Example. In a company, project managers and testing engineers with a security level of at least 5 can access project's documents. The policy in DNF form is $P = [(role \in \{manager, engineer\}) \wedge (security\text{-}level \geq 5)]$. Bob is a project manager from January 1st to January 25th based on a refresh on January 15th. A refresh on January 21st shows his role has changed to testing engineer as of January 20th through March 20th. A refresh on January 15th shows his security level is 6 as of January 10th to March 20th. Another refresh on January 28th reveals security level has been downgraded to 4 since January 26th through March 20th.

Interval Consistency

At this level, it is required to find overlap of freshness intervals (simultaneous freshness) of relevant credentials before the decision time. In the above-mentioned example in Section 3.3.2, suppose Bob requests access to project documents on Jan 18th. Based on refresh results at Jan 15th, decision point finds simultaneous freshness of relevant credentials during Jan 10th-Jan 15th with satisfactory values. So the access will be granted. The stipulated overlap could be found for most recent refresh results of relevant credentials (Figure 3.8-(a)) or by considering both old and new refresh results (Figure 3.8-(b)). In these and subsequent figures, if any refresh is shown on the first line, it returns *Still-Good* while any other refresh returns *New-Value*. Moreover, in all cases the values of the three credentials are *satisfactory*. In Figure 3.8-(a) the overlap is for the most recent

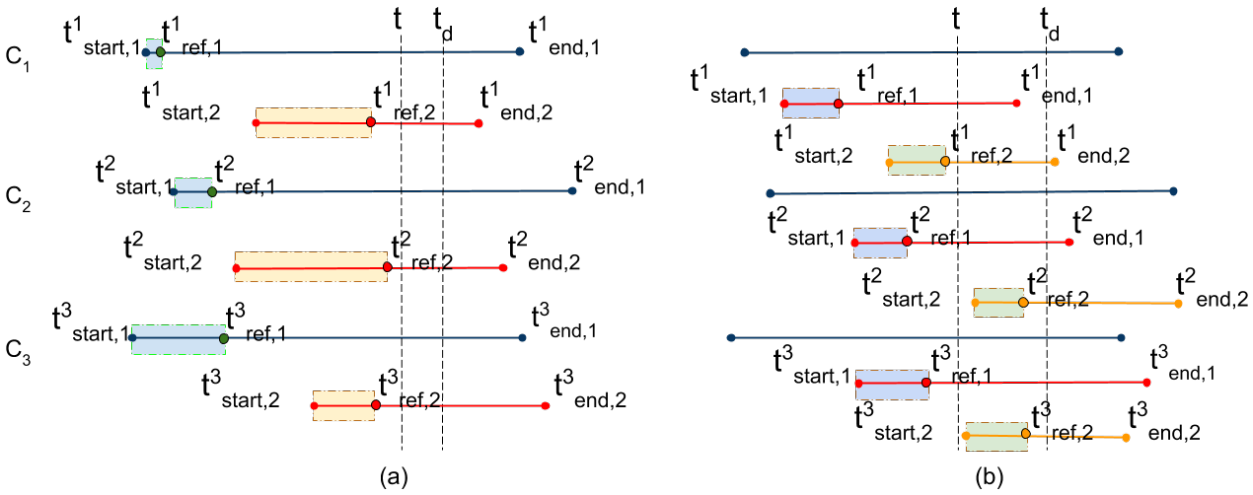

Figure 3.8: Interval Consistency

refreshed values, whereas in Figure 3.8-(b) the overlap is for a mix of the refreshed values, one new and two older.

Specification. Every credential has been refreshed at least once before the decision time and found to be fresh. Most recent values of all relevant credentials are satisfactory with respect to the policy. Any overlap of freshness intervals for the freshest/cached credentials is acceptable so long as the values are satisfactory.

$$
\begin{aligned}
&Interval(V_{DP}^{P,t_d}) \iff (\exists t \leq t_d)(\forall c_i \in V_{DP}^{P,t_d}) \\
&[\max_{\forall c_j \in V_{DP}^{P,t_d}} t^{j}_{start,kmax(t)} \leq t^{i}_{ref,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^{i}_{end,kmax(t)} \\
&\wedge Fresh(c_i, t^{i}_{ref,kmax(t)}) \wedge Fresh(c_i, t^{i}_{ref,kmax(t_d)}) \wedge Sat_{c_i}^{P,t} \wedge Sat_{c_i}^{P,t_d} \\
&\wedge \max_{\forall c_i \in V_{DP}^{P,t_d}} t^{i}_{start,kmax(t_d)} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^{i}_{end,kmax(t_d)}]
\end{aligned}
\tag{3.10}
$$

Property1. There is a time interval during which all relevant credentials were simultaneously fresh with satisfactory values with respect to the policy.

Proof. Based on Equation (3.10), there exists a time (t) prior to the decision time at which the latest refresh of every relevant credential happens after all have been started and before any of them ends. This implies all credentials are simultaneously fresh during $[\max_{\forall c_i \in V_{DP}^{P,t_d}} t^{i}_{start,kmax(t)}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^{i}_{ref,kmax(t)}]$.

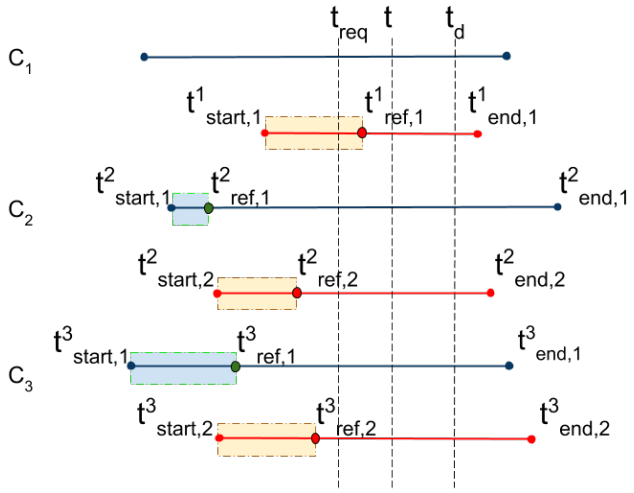

Figure 3.9: Interval Consistency with Request Time

Corollary. if $t = t_d$, the latest values of relevant attributes have freshness overlap.

Comparing with Revocation-Based Scenario

Based on the **Claim** in Section 3.3.1, revocation and refresh are the same in case of *Valid* and *Still-Good* responses from AA. But if the result is *New-Value*, the corresponding revocation result would be *Invalid* which denies the access. In example in Section 3.3.2, if Bob requests access to the project's documents on Jan 25th and decision point rechecks the credentials, although Bob's role has changed, he would get the access in the refresh scenario whereas he would be denied in revocation scenario.

Interval Consistency with Request Time

In the first level, the decision point relies on what avails of previous refresh results for relevant credentials and access would be denied in case of any unrefreshed credential. By considering the request time we could compensate for missing refreshes. In example in Section 3.3.2, if Bob requests for accessing project's documents on January 14th, the access would be denied at first level since there is no refresh result available for required credentials. At second level, the decision point refreshes the credentials after the request time and then checks the consistency requirements. Figure 3.9 shows a similar example where the top credential is refreshed after request time.

Specification. Decision point refreshes any credential with missing refresh results after the request time. Afterwards, relevant credentials should satisfy the interval consistency (previous level) requirements.

$$\begin{aligned} &IntervalWithReq(V_{DP}^{P,t_d}) \iff (\forall c_i \in V_{DP}^{P,t_d})\ [t^i_{ref,kmax(t_{req})} \neq \perp \\ &\vee (\exists t_r\ \ t_{req} < t_r < t_d)\ \ Refresh(c_i, t_r)] \wedge Interval(V_{DP}^{P,t_d}) \end{aligned} \tag{3.11}$$

Proposition. We assume the set of relevant credentials would not change during the short gap between request time and decision time, so, $V_{DP}^{P,t_{req}} = V_{DP}^{P,t_d}$. In other words the policy will not frequently change in the system.

Property1. There is a time interval during which all relevant credentials are simultaneously fresh. Possible lack of refresh would not unnecessarily deny access.

Proof. Use of the same requirement of $Interval(V_{DP}^{P,t_d})$ guarantees the same property of freshness overlap of relevant credentials. Any missing refresh results would be compensated after request time. It is possible that the gap between the request time and decision time does not last enough to compensate for all the lacking information, but we consider it as an administrative setting which is out of scope for this research to quantify.

Property2. Every interval consistent view with request time satisfies the interval consistency requirements as well.

Proof. The proof is trivial since this level is defined based on interval level.

Property3. An interval consistent view may deny access allowed by interval consistent with request time.

Proof. Since we do not consider request time in the first level, there is no opportunity to compensate for possible missing refreshes which could enable access.

Comparison with Revocation-Based Scenario. Considering the formal specification in Equation (3.11), which is based on the first level, the comparison is trivial. If refresh is substituted with revocation, system's availability would decrease. The same situation may happen with regard to the example in Section 3.3.2 as discussed there.

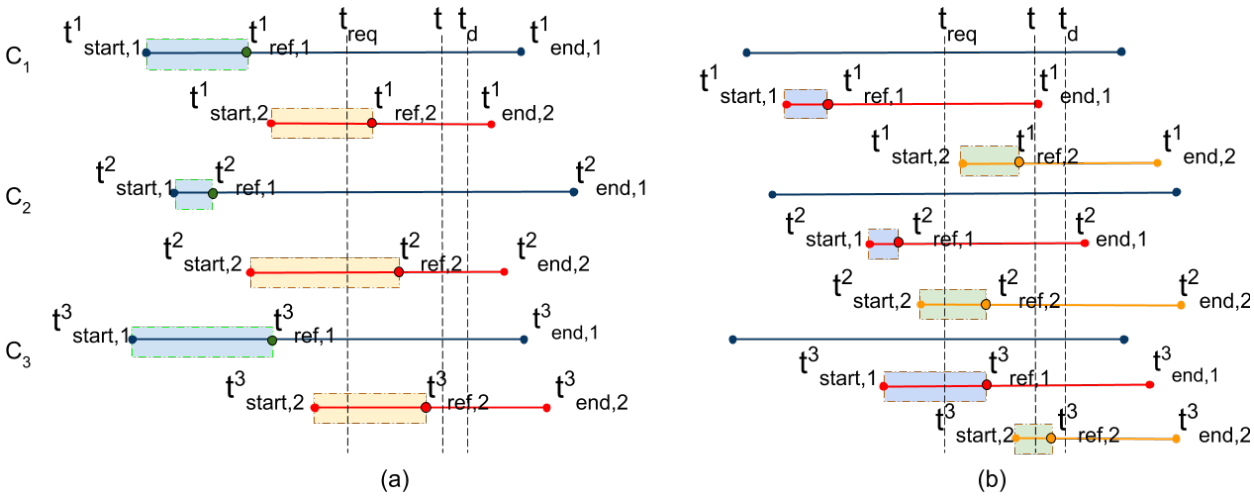

Figure 3.10: Forward Looking consistency

3.3.3 Forward-looking Consistency

This level provides simultaneous freshness of all relevant credentials *after the request time*, considering both new and old credentials. Overlapping interval could either include the request time (Figure 3.10-(a)) or not (Figure 3.10-(b)). In example in Section 3.3.2, if Bob requests access to project's documents on Feb 1st his credentials would be refreshed afterwards revealing changes in role and security level leading to denial. However in previous levels an unauthorized access may be granted.

Specification. Any relevant credential has to be refreshed at least once after the request time. All relevant credentials have to to be found simultaneously fresh at or after the request time.

$$
\begin{aligned}
&ForwardLooking(V_{DP}^{P,t_d}) \iff (\exists t \;\; t_{req} < t \leq t_d)(\forall c_i \in V_{DP}^{P,t_d})[(t_{req} < t^i_{ref,kmax(t)}) \\
&\wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)} \leq t^i_{ref,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t)}) \\
&\wedge Fresh(c_i, t^i_{ref,kmax(t)}) \wedge Fresh(c_i, t^i_{ref,kmax(t_d)}) \wedge Sat_{c_i}^{P,t} \wedge Sat_{c_i}^{P,t_d} \\
&\wedge \max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t_d)} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t_d)}]
\end{aligned}
\tag{3.12}
$$

Property1. There is a time interval during which all relevant credentials are simultaneously fresh after the request time.

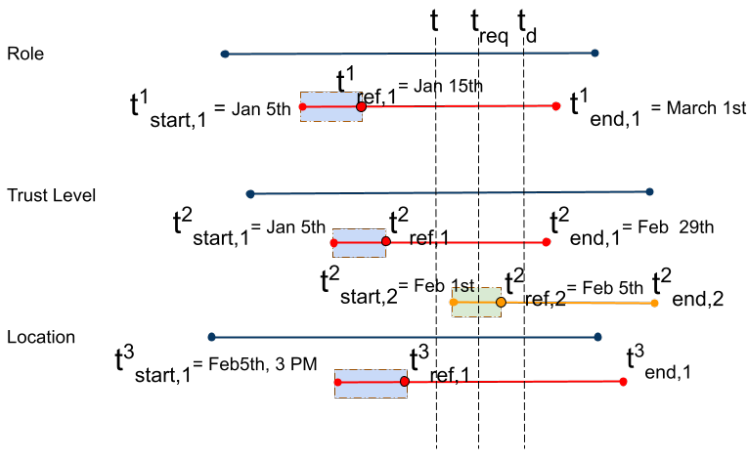

Figure 3.11: Smart Home Use Case: Forward-Looking Freshness Required

Proof. Based on Equation (3.12), all relevant credentials are simultaneously fresh during $[\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{ref,kmax(t)})]$. Part of this interval is located after the request time since refresh has been done after it.

Property2. Every forward-looking consistent view is interval consistent with request time as well.

Proof. Comparing Equation (3.11) and (3.12) shows forward-looking consistency is a restricted version of its preceding level, so the proof is trivial.

Property3. Not every interval consistent with request time view is necessarily forward-looking as well.

Proof. At the second level of consistency, only some credentials need to be refreshed after request time to compensate for lacking information. Whereas in forward-looking consistency, all have to be refreshed after the request time.

Comparison with Revocation-Based Scenario. Changing credentials in revocation scenario leads to hinder the access, whereas in refresh scenario, the *New-Value* in case of any changes would let the subject proceed. In the example mentioned in Section 3.3.2, Bob's request to access project's documents on Jan 20th would be denied in a revocation-based scenario, however in refresh scenario access would be granted.

3.3.4 Smart home Use Case

Imagine an ABAC policy in a smart home IoT environment which regulates access to house's thermostat functionalities. The rules in this policy regulate user-to-device access based on user's attributes of *age*, *role*, *location*, and *trust_level* on one hand, on the other hand there is a *required trust level* for each of thermostat's functionalities. One of the requisites to access a functionality for a user trying to acquire it, is having a *trust_level* greater-equal than assigned *trust_level* to that specific functionality. So, the homeowner (or any home administrator) can assign highest trust_level for scheduling thermostat, while users like kids, babysitter or any user other than parents would not be assigned such a high trust level. Therefore, the thermostat's scheduling functionality would be only available to parents. Consider following simplified policy as a sample:

$$(s : user, o : thermostat, op : operation) \iff ((op.trustLevel \leq u.trustLevel) \wedge ((u.role = "parent") \vee (role \in \{babysitter, kid\} \wedge (u.location = "home"))))$$

Based on the above policy, when a user u requests access to functionality op of thermostat, three user attributes would be checked by the decision point to make an access decision, either grant or deny. These attributes are supposedly delivered to the decision point through an attribute credential. As indicated in the above policy, each functionality (a.k.a op) of the thermostat has its own assigned trust level. Suppose the trust level to be a number between 1 and 5. For turning the thermostat on/off this number should be at least 2, and for scheduling it should be 5. Therefore, based on the policy, if a parent with $trust\_level = 5$, wants to do any operation on thermostat it would be granted. However a kid or a babysitter are authorized to do only on/off operation while at home and their assigned trust level matches the requirements. Now consider the following example.

Example. Alice is a newly hired babysitter. On Jan 5th, when she started her job, she was assigned the lowest trust level, say 1. So, based on the above policy she has no access to any of the thermostat's operations. After a few days on Feb 1st, the parent updates Alice's trust level value to 2. Now, she can turn the thermostat on/off. If she requests access to turn on/off thermostat on Feb 5th following is what happens. Imagine the revocation-based scenario, if any of the

levels of previously mentioned consistency based on revocation is applied by the decision point (see Section 3.2.2), Alice's access to turn the thermostat on/off would be denied, even for the forward-looking consistency in which the revocation check happens after the request time for all credentials. The reason is the revocation check only shows the previous credential which contained $trust\_level = 1$ is revoked and no new value would be returned. However, relying on refresh-based consistency level the story can continue as follows. After updating Alice's trust level if she requests access to turn on/off functionality of thermostat, instead of *revoked* status, the *new_value* would be returned which grants access to Alice. Now suppose, Alice maliciously wants to turn off the thermostat after she leaves the house. Although she is not home anymore, relying on any level rather than *freshness-forward looking consistency* would authorize her access, while it should be denied. The reason it is only at *forward-looking* level that we need the freshness interval of all credentials to overlap. This situation has been demonstrated in Figure 3.11.

3.4 Safety and Consistency of Mutable Attributes Using Quotas: A Formal Analysis

Previous works have assumed attributes to be immutable, to wit their values could be changed only via administrative actions. However, so far there is no research carried out in the context of mutable attributes, values of which could be changed as a result of users' access. Our central goal in this research is to investigate the safety and consistency problem in the context of mutable attributes. Mutability adds further complication to establish consistency requirements and specifications, as it requires synchronization mechanisms in place to update attribute values. There is a rich body of literature in distributed environments dealing with concurrency. However, practical solutions are typically dependent on the specific application domain. We identify two categories of use cases of practical benefit in the context of ABAC, which turn out to be amenable to quota-based solutions. More general treatment of consistency beyond quota-based solutions is beyond the scope of this dissertation. We develop a formal characterization of required consistency using *refresh* in this context. We also observe that revocation is inappropriate to be used in the context of mutable

attributes, while it has been the traditional approach to attribute freshness for immutable attributes.

3.4.1 Problem Statement and System Assumptions

Assuming an ABAC model is in place, we discuss the safety and consistency problem for mutable attributes in this research. We assume attribute credentials are provided through different attribute authorities (AA) and there is a single decision point in the system. The main goal is to propose a practical approach to limit the exposure of the decision point to outdated attribute values. Administrative changes of attribute values are always done at the AA for both mutable and immutable attributes. It is worth reminding that we consider attributes mutable if their changes are the consequence of access utilization by subjects. So, if any administrative change happens that should be managed administratively. As an example if the user's credit line changes, that change has to be managed and take effect via AA whereas using the credit line to purchase services is done automatically by the system. It is also possible to check the most recent values of mutable attributes with AA for each utilization, for example after each credit card utilization. However, it makes AA a single point of failure and is as inefficient as any centralized approach. There is a big space of analysis to deal with consistency problems in distributed environments with mutable attributes. Our analysis focuses on a quota-based approach in which every mutable attribute value would be treated as a quota and AA would delegate the quota to some predetermined distributed servers and those servers take care of utilization of delegated quotas and update them locally.

Quota-Based Approach

Quota for reusable resources could be considered as reimbursable deposit which is refunded after usage has finished [144]. There is another type of resource known as consumable which would decrease in amount at each access without being refunded. A system that allows a user up to five concurrent sessions for content streaming is an example of the former, since termination of a session enables another one to be initiated. A system that allows a total of five uses is an example of the latter; once consumed, the quota would not be refunded. Access quota has been previously

used in risk-based access control. Authors in [138] assigned quota to users and obligations in order to regulate access risks. Access quota has been used in [203] to specify a threshold on tolerable risk by the system through assigning quotas based on estimation of access needs during a specific period of time.

We assume each mutable attribute has a global limit known to the corresponding AA. This global limit can be managed in a centralized way by AA to be distributed and managed among users whose access requires utilizing that attribute. In another approach, the global limit could be delegated to local servers which would be responsible to distribute the global limit as local quotas to be manipulated through different access utilizations. The totality of local quotas distributed in this manner cannot exceed the global quota. In any case, we recognize two approaches to apportion the global limit as follows.

1. Service-Based: In this approach AA assigns portions of the global limit to each service point, regardless of the users who are utilizing access to the service. So, a global limit would be set on concurrent number of service usage by all users. It is notable that a global limit would be set for each service instance of a service. As an example, an Internet Service Provider supplies internet connection to hotel rooms. Each room's internet connection would be a service instance. Regardless of which users/devices are connected to the service, an overall internet usage limit would be set for all provided service instances.

2. User-Based: In this approach a subscriber (user) to a service would be assigned a limited number of concurrent sessions to use the service. As an example, assign a specific amount of storage to every user on the cloud or put an upper limit on the number of concurrent sessions that each subscriber to a TV channel can have. Note that attribute-based access control could be done either id-less (anonymous) as proposed in Idemix [46] or non-anonymous as we address it in user-based global limit assignment.

In both approaches a global count limit would determine the upper bound of usage which restrains the concurrent usage number of the service objects. Upper bound could be set either as a count-

down which is consumable and non-refundable after each usage or it can be restored after access utilization has been completed.

Definitions and Assumptions

We examine the safety and consistency problem from the perspective of a single access decision point within a larger distributed ABAC authorization system (which would include multiple such decision points amongst other distributed components). Attributes of objects and environment and the policy are presumed to be known with high assurance to the decision point. Our focus is on subject attributes in this dissertation. As previously discussed, subject attributes could be either mutable or immutable. The decision point is the entity which checks the policy and determines the set of attributes whose values should satisfy policy requirements. We call this set *relevant attributes set* or *relevant attributes* for short, following previous studies [116, 183, 185], which could be distributed over multiple attribute providers. The set of relevant attributes might change over time as the decision point examines the policy expression which we assume is expressed in Disjunctive Normal Form. If an attribute's value does not fulfil policy requirements, it would be replaced with the next conjunct in the same clause, so the set of relevant attributes changes.

Definition 4. We call the set of relevant attributes to the policy P at time t determined by decision point DP, the *view* of the decision point at time t and denote it as $V_{DP}^{P,t}$.

3.4.2 Use Case Scenarios

In this section we discuss two sets of practical use case scenarios to illustrate how utilizing the conferred access may require updating mutable subject attributes. These use cases are provided in the context of ABAC and are amenable to quota-based approaches. Throughout use case explanations we use a quota-based solution to manage concurrent accesses of users to service instances which is a well-established approach to handle concurrency in distributed environments. We consider pre-authorization and pre-obligation models with respect to UCON. In order to explain each use case we follow UCON notation of [150]. Moreover, we define the following sym-

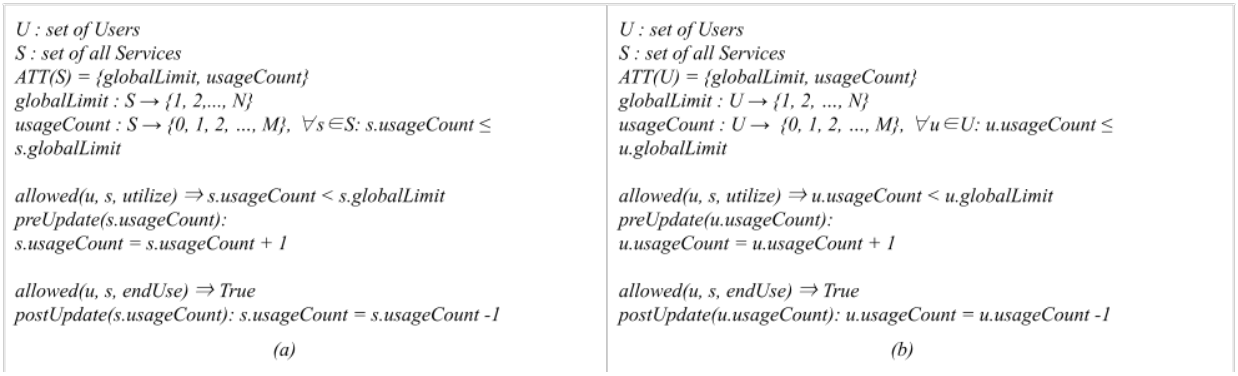

Figure 3.12: Centralized Approach to Manage Global Limit: a) service-based b) user-based

bols to describe our use cases. U and S represent the set of all users and services in the system respectively. Each concurrent session of the user is shown as a User Instance (UI) (a.k.a subject). Considering the service as the object in our system, we represent each service instance as a Service Instance Object (SIO). The number of existing sessions is treated as a user's attribute which ought to appropriately change as new sessions are created or terminated while using the service. $ATT(.)$ indicates the set of attributes for the entity enclosed in parenthesis. $X.Y$ denotes attribute Y of the entity X. $preUpdate(.)$ and $postUpdate(.)$ indicate functions which have to be done to update attribute values before usage is started and after usage is terminated respectively. The notation $allowed(s, o, a) \Rightarrow$ indicates necessary requirements for subject s to be allowed to do action a on object o.

Centralized Approach

AA distributes attribute values among entities to be used during access evaluation and utilization, so dissemination is done in a centralized way under a single authority. This strategy makes the AA as the single point of failure which monitors and tracks assigned limits. Attribution of the assigned portion could be done for all users of a specific service, a.k.a service-based, or it could be user-centric, a.k.a user-based, as follows. Either of the following two approaches could also be changed to be processed as a global countdown limit, which means the global limit would be consumable and would not be refunded after each usage.

1. **Service-Based Distribution:** In this type of distribution a global limit is assigned for all users of a service. The AA would manage sharing the service among different users scattered throughout local or distributed locations. Formal specification of this method is shown in Figure 3.12(a). As an example, an Internet Service Provider supplies internet connection to hotel rooms. Regardless of which users/devices are connected to the internet, the central server would set a usage global limit on the service it provides to the hotel.

2. **User-Based Distribution:** This strategy disseminates the global limit between different users of a service. The global limit can determine the maximum number of service usage per user. As an example a subscriber to a service could have up to M concurrent sessions for using that service. Each time a user wants to start/end utilization of the service, AA would check the prerequisites (if any) and update mutable attributes accordingly. Formal specification is given in Figure 3.12(b).

Distributed Quota-Based Approach

Although a centralized approach provides the benefit of avoiding inconsistency because of its central management, it presents scalability and fault tolerance difficulties. To provide a system with better fault tolerance and to avoid AA from becoming the single point of failure, a distributed approach could be exploited. In this approach, a global limit which has been assigned to every mutable attribute would be distributed among some local servers which delegate their assigned share as *quotas* to each service/user throughout access assessment and practice. Furthermore, this limit could be allocated using a service-centric or user-centric approach as follows.

1. **Service-Based Quota Distribution:** A global limit would be set per service to be used by different users. This limit would be delegated to different service providers in the distributed environment which then could be allocated to different users of the system per request. As an example, consider a Software as a Service (SaaS) [201] which provides all users from a university's IP address up to a total of M concurrent sessions to use the software. Then the global limit could be further distributed among Service Instance Objects (SIO) as quotas to

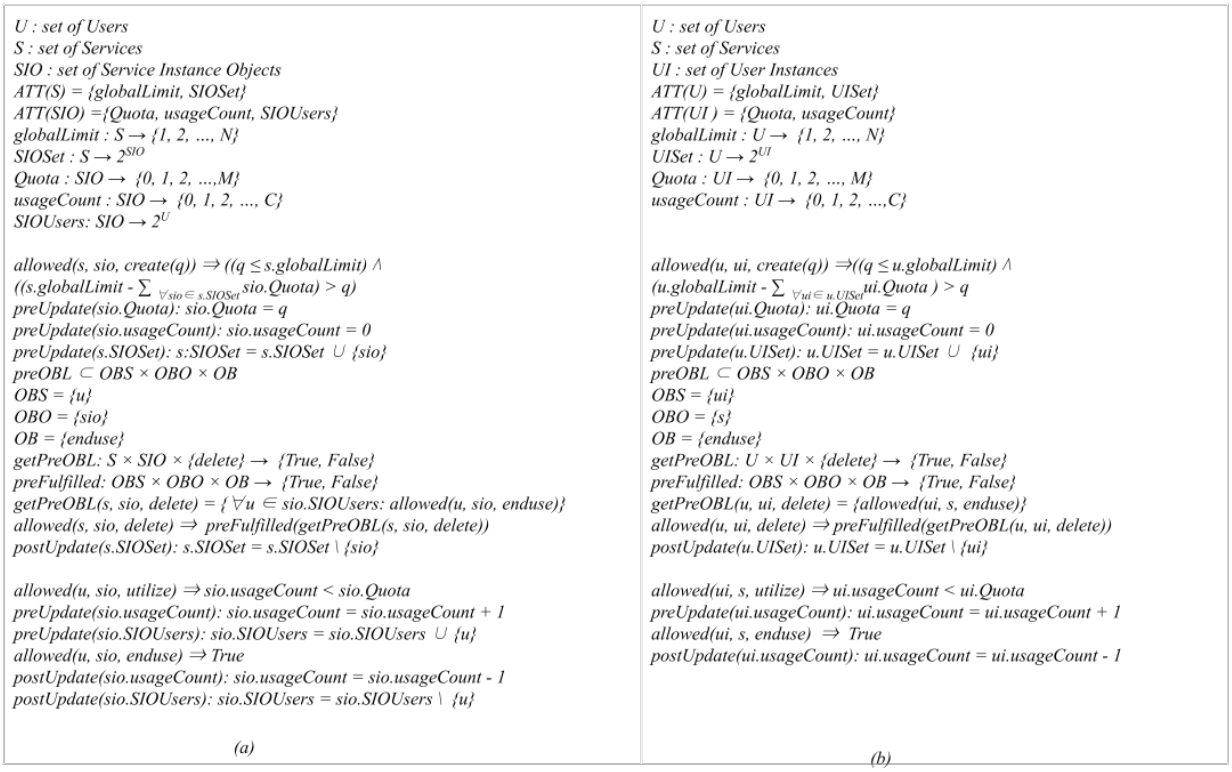

Figure 3.13: Distributed Approach to Manage Global Limit: a) service-based b) user-based

be assigned to different colleges and departments which could be managed locally. Formal specification is provided in Figure 3.13(a).

2. **User-Based Quota Distribution:** In this approach a global limit would be allocated to local servers for each user. The global limit determines the maximum number of simultaneous service usage a user can have. A user (U) can have multiple concurrent user instances (UI) which is created with a predetermined share ($ui.Quota$) of its parent's (the user who created that UI) global limit. The sum of all assigned quotas to different user instances cannot exceed their parent global limit. As an instance, a subscriber to a TV channel can have up to 5 concurrent live sessions and then this global limit can be used on different devices to watch that channel. Formal specification is provided in Figure 3.13(b).

To delete one of the user's user instances, we enforce one of the two following approaches to ensure the assigned quota to deleted user instances would be set free for further utilization.

- user instance which the user wants to be deleted, should have no service utilization

($ui.usageCount = 0$).

- terminate all services which are being practiced by to-be-deleted user instances. To satisfy this requirement, we enforce an obligation to be fulfilled by the user to first discontinue all user instance services and then proceed to delete the user instance. This approach has been used in Figure 3.13(b).

The quota upper limit could also be set as a countdown limit which conveys that the quota is non-refundable after usage termination.

Distributed vs. Centralized Quota Management

Each of centralized and distributed quota management approaches discussed above has its own advantages and drawbacks. In the rest of this section, we provide some properties of each method of quota management to compare their pros and cons.

Property1. Centralized quota management provides correct access control decisions.

Proof. In the centralized approach, all required attribute credentials are kept in one place which is a highly assured AA. As long as there is no network failure and post updates could succeed, this would result in granting access only when it is correct based upon the policy. Nevertheless, underlying information could be always incorrect or attacked, but we consider that out of scope of this research.

Property2. Distributed quota management approach provides less availability and less utilization, compared to the centralized approach.

Proof. It is possible to block an access in the distributed approach due to lack of available quota, while it would be conferred in the centralized approach. In a distributed approach, the global limit of every attribute is dispersed between local distributors (servers) as quotas. If an access permission requires assessing a specific attribute, access would be granted provided a spare quota is available. If not, access would be denied. This could happen even if there are some spare quotas for the same attribute sitting unused on other servers. If this request was delivered to a system with a centralized quota management, access would not be denied as long as there is any available spare quota and it

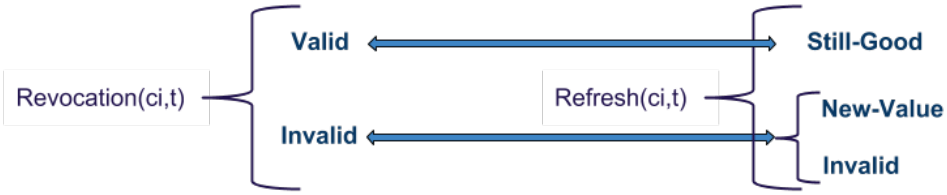

Figure 3.14: Revocation vs. Refresh [183]

would be allocated as the global limit is managed centrally by the AA. So, even while post updates succeed, distributed approach could be deficient in availability and resource utilization compared to centralized approach.

Property3. Distributed quota management access provision is correct.

Proof. Based on Property2, a distributed approach would provide less availability compared with a centralized approach. This conveys less access would be granted while applying a distributed approach. In other words, granted accesses in a distributed method is a subset of accesses granted in centralized one, which are correct based on Property1.

3.4.3 Consistency Levels for Distributed Quota-Based Distribution Methods

In this section we only look at the refresh-based solution as revocation is not applicable that which we discuss in Subsection 3.4.3. Subsection 3.4.3 accentuates the fact that the consistency problem would arise only when multiple (more than one) attributes are included in the view of decision point. Two consistency levels in Subsection 3.4.4 are provided to reduce decision point exposure to outdated values while its view contains both mutable and immutable attributes, recognizing that mutable attributes would increase the risk of exposure to stale values, a.k.a safety and consistency problem.

Revocation vs. Refresh

Authors in [183] compared the two possible ways to obtain latest attribute values as shown in Figure 3.14. While in revocation it is only possible to check if attribute credential is either valid or revoked, in the refresh scenario new values of attributes could be returned in case of any changes

other than revocation. So instead of only invalidating the old value, the new value would be communicated to the requesting party. Taking mutability into account, the decision point needs to be updated by recent values of relevant attributes. Since revocation check only evaluates the validation of previous attribute values, it is appropriate only for immutable attributes which solely could be updated by administrative actions. If used for mutable attributes, revocation check is useful only if the attribute value has been revoked, but any changes in the value would not be reflected in revocation check response. Both revocation and refresh scenarios are considered as pulling approaches, in which the recent attribute value information will be recovered via querying AA. We consider the refresh scenario to be appropriate to obtain the freshest values of attributes in this work. However, if the global limit changes, there should be a pushing mechanism from AA to distributing servers to make them aware of the change. Consideration of this latter mechanism is out of scope for this research.

Consistency Considerations for Mutable Attributes

Consistency problem arises when the decision point needs more than one attribute value to make an access decision. We call the set of required attributes to make an access decision *relevant attributes* as specified in Definition 4. Practical use cases compliant to quota-based approach have been discussed in Section 3.4.2, all of which consider only one mutable attribute. When relevant attributes include more than one attribute, there is always the risk of some attribute values being outdated while the decision point is trying to acquire other attributes' values from distributed attribute authorities. Previous research has been done toward definition of different consistency levels by imposing restrictions on timeliness of attribute checks [183, 185], but all attributes presumed to be immutable. The following example demonstrates the consistency problem when the set of relevant attributes include more than one attribute which also includes a mutable attribute as well.

Example. A user tries to create a backup of his phone contents on Apple iCloud. To grant access to iCloud storage, the decision point needs to confirm the validity of Apple ID which is considered

Table 3.3: Summary Table of Symbols

Symbol	Meaning
t_{req}	request time
t_d	decision time
c_i^m	i^{th}credential which is mutable
c_i^{im}	i^{th}credential which is immutable
c_i	i^{th}credential, regardless of being mutable/immutable
$t^i_{ref,k}$	time of k-th refresh of c_i^{im}
$t^i_{update,k}$	time of k-th refresh of c_i^m
$t^i_{start,k}$	attribute start time of c_i after k-th refresh
$t^i_{end,k}$	attribute expiration time of c_i after k-th refresh
$kmax(t)$	latest refresh of c_i before time t (c_i is determined by context)
$val^i_{kmax(t)}$	the value of c_i after $kmax(t)$-th refresh
$t^i_{ref,kmax(t)}$	time of $kmax(t)$-th refresh of c_i^{im}
$t^i_{update,kmax(t)}$	time of $kmax(t)$-th refresh of c_i^m
$t^i_{start,kmax(t)}$	attribute start time of c_i after $kmax(t)$-th refresh
$t^i_{end,kmax(t)}$	attribute expiration time of c_i after $kmax(t)$-th refresh

as an immutable attribute as well as remaining storage assigned to that ID as a mutable attribute. Suppose a scenario in which the Apple ID has been validated previously and the decision point tries to check the remaining iCloud storage. It is possible that Apple ID has been invalidated while trying to acquire remaining storage, which indicates an inconsistency situation. The reverse order of checks is also possible to cause the consistency problem where iCloud storage has been consumed up with content from other devices connected to the same Apple ID, while checking the authenticity of the ID.

3.4.4 Formal Specification of Consistency Levels

Before defining formal consistency levels, we emphasize that mutable attributes would be updated more frequently than immutable ones. As a justification, we remind the reader of the definition of each category of attributes. Immutable attributes are assigned and only could be changed by administrative actions, however mutable attribute values could change as a side effect of each access utilization. Since any access could change the relevant mutable attributes values, it is reasonable to

assume mutable attribute changes more frequently. Based on the previous statement, unlike what has been defined in previous works to define consistency levels based on different recommended freshness/validity overlaps, we would not assume the same degree of freedom for system administrators to decide lower levels of freshness overlap to be provided to the decision point. So, at least all of the mutable attributes have to be refreshed after the request is submitted to the decision point.

It is notable that we consider the request time as the anchor point in that it is the closest recognizable point in time to the decision time and we want to check the value of mutable attributes at the closest possible point to the decision time. Nonetheless for immutable attributes we could rely on refresh results which have been done before the request time. In contrast to mutable attribute values which are available locally at local distributors in a distributed approach and so could be updated at any arbitrary time, immutable attributes updates require the decision point to consult the AA which might not be possible at any desired time in distributed environments. That said we can give more freedom about timeliness of immutable attributes. However, simultaneous freshness of mutable and immutable attributes might not necessarily be provided. We propose two levels of consistency assuming the set of relevant attributes includes both mutable and immutable attributes as the most general case. Following [183], Table 3.3 shows required symbols with a brief explanation of each.

Lifetime Overlap Level

This level of consistency guarantees that all relevant credentials would be overlapping in their lifetimes. It also provides the decision point with the freshest value of relevant mutable attributes at the decision time. However immutable attributes freshness cannot be assured as it may not be possible to refresh them after the request time. So, immutable attribute values might be outdated but correct in the past. Decision point would rely on values of latest available refresh results for immutable attribute credentials whenever refresh after request is unfeasible. Yet, refreshing mutable attribute credentials after request time is indispensable, in that their values could have been altered since last refresh as the usage side effect.

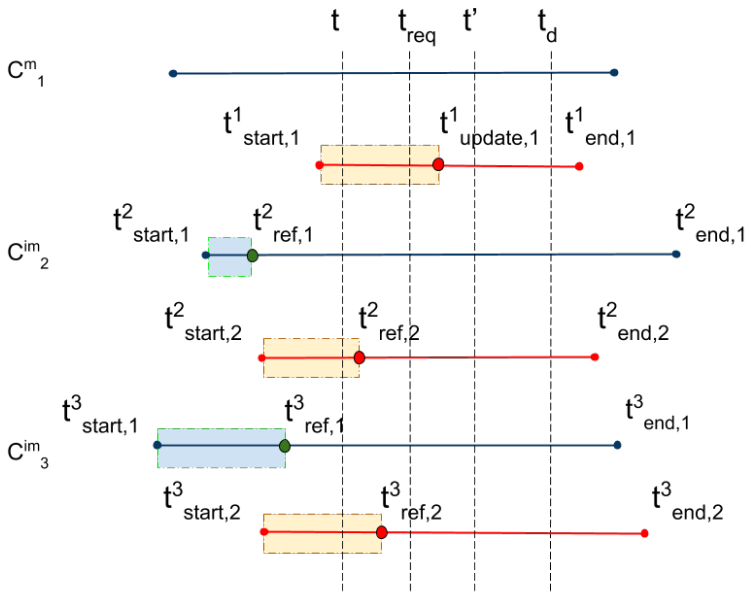

Figure 3.15: Lifetime Overlap Consistency Level

The decision point in Figure 3.15 relies on three attributes, the first of which is mutable and two others are immutable. As depicted the mutable attribute has been refreshed after the access request, however for mutable attributes the latest refresh results, which have been acquired before the request time, have been used.Formal specification of this level is as follows.

Specification. Every immutable attribute has to be refreshed at least once before the decision time and found to be fresh. Every mutable attribute has to be refreshed at least once after the request time and before the decision time and found fresh based on the latest refresh results.

$$
\begin{aligned}
&LifetimeOverlap(V_{DP}^{P,t_d}) \iff (\forall c_i^{im} \in V_{DP}^{P,t_d})(\exists t \;\; t \leq t_d) \\
&[(\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)} \leq t^i_{ref,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t)}) \wedge Fresh(c_i, t^i_{ref,kmax(t)})] \\
&\wedge (\forall c_i^{m} \in V_{DP}^{P,t_d})(\exists t' \;\; t_{req} \leq t' \leq t_d) \\
&[(\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t')} \leq t_{req} \leq t^i_{update,kmax(t')} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t')}) \wedge Fresh(c_i^m, t^i_{update,kmax(t_d)})] \\
&\wedge \max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t_d)} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t_d)}
\end{aligned}
\tag{3.13}
$$

Property1. All relevant attributes lifetime intervals would overlap.

Proof. Based on Equation 3.13, all credentials would be refreshed when other credentials are

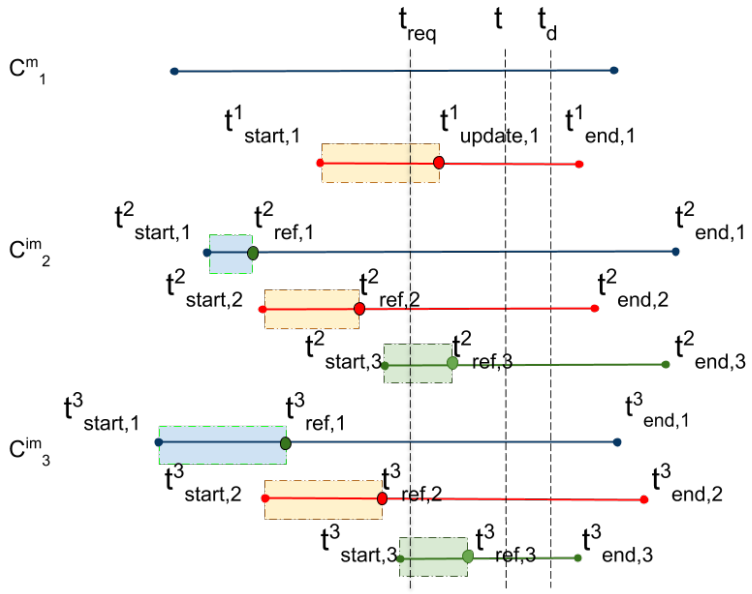

Figure 3.16: Freshness Overlap Consistency Level

in their lifetimes. Moreover, decision time lies in the last known lifetime interval for all relevant credentials at the decision time, as stated in the last part of Equation 3.13. This conveys at least one point (t_d) lies in intersection of all credentials lifetimes. So, all lifetimes would overlap in $[\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t_d)}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t_d)}]$. Although there is no guarantee for simultaneous freshness of all relevant credentials, Figure 3.15 depicts a lucky situation in which all credentials are fresh during $[t^1_{start,1}, t^2_{ref,2}]$. But even in this case there is no simultaneous freshness of all attributes after the request time.

Freshness Overlap Level

This level of consistency guarantees that all relevant credentials would be fresh simultaneously after the access request turned into the system. It requires all relevant credentials, including both mutable and immutable, to be refreshed after request time. As depicted in Figure 3.16, three attributes have been considered to be relevant, first of which is mutable and two others are immutable. All three relevant attributes have been refreshed after the request time which supplies the decision point with the freshest values of each relevant credential while all freshness intervals are guaranteed to overlap after the request time. Following is the formal specification of this level.

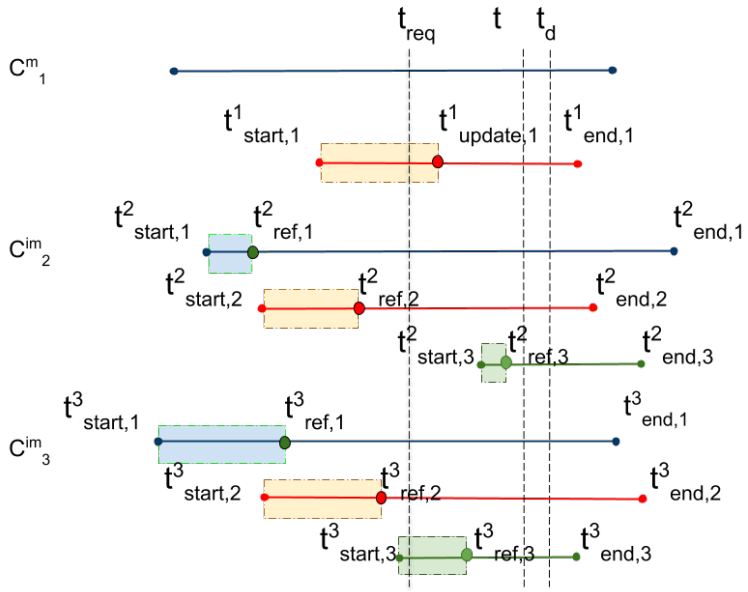

Figure 3.17: Start Time Has Fallen After Request Time

Specification. Every credential has to be refreshed after request time and before the decision time. It is required for all relevant credentials to be started at or before the request time. So, simultaneous freshness of all relevant attributes is guaranteed. If some credentials start time fall after the request time, both mutable and immutable attributes would be assured to be fresh at some time interval after the request time but simultaneousness of freshness intervals is not assured. So, we restrict the start times to fall at/before request time.

$$
\begin{aligned}
&FreshnessOverlap(V_{DP}^{P,t_d}) \iff (\exists t \;\; t_{req} < t \leq t_d) \\
&[(\forall c_i^{im} \in V_{DP}^{P,t_d})(t^i_{start,kmax(t)} \leq t_{req} < t^i_{ref,kmax(t)}) \\
&\wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)} \leq t^i_{ref,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t)}) \wedge Fresh(c_i^{im}, t^i_{ref,kmax(t)})] \\
&[(\forall c_i^{m} \in V_{DP}^{P,t_d})(t^i_{start,kmax(t)} \leq t_{req} < t^i_{update,kmax(t)}) \\
&\wedge (\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)} \leq t^i_{update,kmax(t)} < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t)}) \wedge Fresh(c_i^{m}, t^i_{update,kmax(t_d)})] \\
&\wedge \max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t_d)} < t_d < \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{end,kmax(t_d)}
\end{aligned}
\tag{3.14}
$$

Property1. All relevant credentials would be simultaneously fresh during a time interval before the decision time which includes the request time.

Proof. All credentials have to be refreshed after request time and before decision time. Also the latest start time of all credentials should happen before/at the request time. So all credentials would be simultaneously fresh during $[\max_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{start,kmax(t)}, \min_{\forall c_i \in V_{DP}^{P,t_d}} t^i_{ref,kmax(t)}]$ time interval which includes the request time.

Property2. Every view at Freshness Overlap level would be at Lifetime Overlap level as well.

Proof. This property is obvious, since the freshness interval for every credential is a sub-interval of its lifetime interval. Therefore, when there is freshness overlap, lifetime overlap is self-evident.

Property3. Not every view at Lifetime Overlap level is at Freshness Overlap level as well.

Proof. As presented before, at the Lifetime Overlap level it is possible to have some immutable attributes with refresh time even before the request time. On the contrary, to be at Freshness Overlap level, every immutable credential requires to be updated at least once after the request time. Thus although the lifetimes would overlap based on the last condition stated in Eq. 3.13, freshness overlap could not be guaranteed.

Tip. It is significant to note that enforcing the start time of refreshed attributes to lie before the request time ($t^i_{start,kmax(t)} \leq t_{req} < t^i_{ref,kmax(t)}$) is the key requirement in this level. Otherwise it is possible to have non-overlapping freshness intervals although all credentials have been found fresh after the request time. For example, as seen in Figure 3.17, latest start time of c_2^{im}has fallen after request time ($t^2_{start,3} > t_{req}$) and its freshness does not overlap with freshness interval of other two credentials.

3.4.5 Smart Home Use Case

Suppose in a smart home IoT parents want to limit the access of each kid to PlayStation to weekends, not exceeding 120 minutes (2 hours). Continuous control over the amount of time is not possible unless considering the playtime upper bound as a *mutable* attribute, because we expect playing time to be subtracted from the set upper bound. The upper bound would be set administratively by parents, however it could be changed as kids utilize their playtime, say as the side effect of user's usage. Following is a simple attribute-based policy example:

Table 3.4: Centralized User-Based Quota Management: Smart Home Use Case

U, S : set of Users and Services repectively. $\mapsto U = Alex, S : \{PlayStationRemotePlay\}$
UI: set of User Instances. $\mapsto$ UI: set of account activations on different devices
$ATT(U) = \{globalLimit, UISet\}$
$ATT(UI) = \{Quota, usageCount\}$
$globalLimit : U \rightarrow \{1, 2, ..., N\} \mapsto globalLimit : Alex \rightarrow \{1, 2, ..., 120\}$
$UISet : U \rightarrow 2^{UI} \mapsto UISet = \{accountsonmacbook, laptop, PS4\}$
$Quota : UI \rightarrow \{0, 1, 2, ..., M\} \mapsto$ each account on a device can have its quota
$usageCount : UI \rightarrow \{0, 1, 2, ..., C\} \mapsto usageCount = \{0, 1, ..., playtime\}$
$allowed(u, u_i, create(q)) \rightarrow ((q \leq u.globalLimit) \wedge (u.globalLimit - \Sigma_{\forall u_i \in u.UISet} u_i.Quota) > q$
$preUpdate(ui.Quota) : ui.Quota = q$
$preUpdate(ui.usageCount) : ui.usageCount = 0$
$preUpdate(u.UISet) : u.UISet = u.UISet \cup \{u_i\}$
$preOBL \subset OBS \times OBO \times OB$
$OBS = \{u_i\}, OBO = \{s\}, OB = \{enduse\}$
$getPreOBL : U \times UI \times delete \rightarrow \{True, False\}$
$preFulfilled : OBS \times OBO \times OB \rightarrow \{True, False\}$
$getPreOBL(u, u_i, delete) = allowed(u_i, s, enduse)$
$allowed(u, u_i, delete) \rightarrow preFulfilled(getPreOBL(u, u_i, delete))$
$postUpdate(u.UISet) : u.UISet = u.UISet \setminus u_i$
$allowed(u_i, s, utilize) \rightarrow u_i.usageCount < u_i.Quota$
$preUpdate(u_i.usageCount) : ui.usageCount = ui.usageCount + 1$
$allowed(u_i, s, enduse) \rightarrow True$
$postUpdate(u_i.usageCount) : ui.usageCount = ui.usageCount - 1$

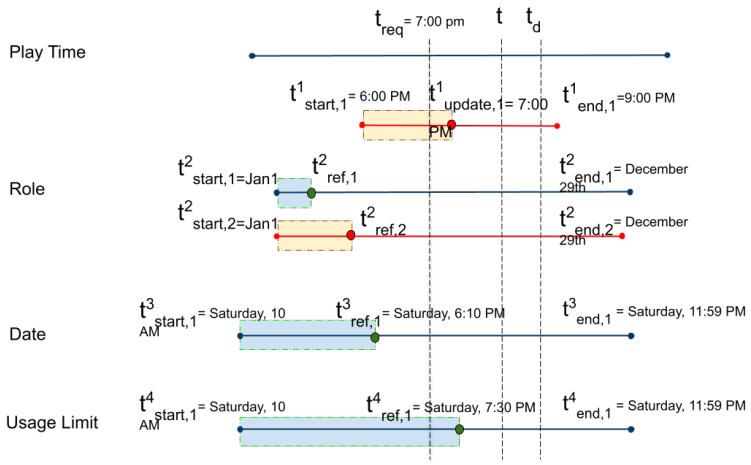

Figure 3.18: Smart Home Use Case: Freshness Overlap Required

$(s : user, o : playstation, op : operation) \iff (u.quota > 0) \wedge ((u.role = "parent") \vee (u.role = "kid")) \wedge day\_of\_week \in \{Saturday, Sunday\}$

In this example we considered the quota distribution to be done as centralized, as mentioned in Section 3.4.2. Since parents are considered to be the only home administrators, utilizing the *centralized* approach looks common-sense. Moreover, we opt for *user-based*, as parents aim for setting the upper bound of playtime for each user, say kids. Formal specification of this approach has been represented in Table 3.4 in which each item has been mapped into its correspondent in our use case.

To determine which level of consistency is required to be imposed by the decision point consider the following example. Imagine Alex, who has the kid role, wants to continue playing on Monday. If the decision point opt for *lifetime overlap* level, and because of intermittent connection cannot update its calendar, it may rely on previously cached information and let the kid play. However, the *freshness overlap* level requires all credentials to have an overlapping freshness interval. Therefore, in case of intermittent connection the decision point cannot proceed. If the connection problem resolves, because of the desired level of consistency, the decision point would realize the day of the week and deny Alex access, as it should do. This situation has been represented in Figure 3.18.

3.5 Discussion: Model Properties and Limitations

In this chapter we formulated different levels of consistency in distributed multi-authority attribute-based access control environments. This section discusses some properties and restrictions of presented consistency levels in each context.

3.5.1 Safety and Consistency of Subject Attributes for Attribute-Based Pre-Authorization Systems

Implementation Implications

Higher level of consistency requires additional checks. There is a tradeoff between the safety assurance provided by higher levels and cost of additional checks, as follows.

- From incremental/internal to r-incremental: the end times relate to decision time, so additional check of end time is required at the decision point in r-incremental.
- From r-incremental to interval: it potentially requires additional revocation checks, because all relevant credentials have to be checked at least once for their latest revocation status after all credentials have been started.
- From interval to forward-looking: all credentials definitely need to be checked for revocation status after the request time.

Quantitative performance evaluation is beyond the scope of this paper. It would require concrete system and workload assumptions and would be specific to the particular context.

Short-lived Credentials

Short-lived credentials are used to obviate the need for revocation check by keeping credential lifetime very small. For our purpose we assume there is an implicit revocation check at start time, otherwise the AA would not issue the credential. No further revocation check is possible. In this case r-incremental and interval consistency will be equivalent. Forward-looking consistency could

be guaranteed only if the request time has pushed prior to the start time for all credentials. The practical implication is that the decision point would need to assemble required subject credentials from appropriate AAs after the request time.

Considering Enforcement Time

After the decision point makes the access decision, it will be enforced by an enforcement point which could be the same or a different entity than the decision point. We certainly know that $t_d < t_e$. Proposed consistency levels in this paper remain unaffected by taking enforcement time into account. From another standpoint, if there is a large gap between the decision and enforcement time, it is possible to utilize an access while some of the corresponding credentials have expired; this is more probable in case of short-lived credentials. So, we can add more constraints to consistency level specifications which restricts this gap as follows: $t_e \leq \min_{\forall c_i \in V_{DP}^{P,t_d}} t_{end}^i$. Therefore, enforcement time could be considered to extend proposed levels of consistency.

3.5.2 Refresh Instead of Revoke Enhances Safety and Availability

We presented three levels of consistency, where each higher level provides enhanced availability and safety at the cost of refreshing more frequently. We compared the qualitative benefits of each level. Quantifying cost-benefit is highly implementation and application specific, and is beyond the scope of this research. Furthermore, there are issues related to managing the risks inherent to applying ABAC in a distributed environment, since ABAC introduces new challenges in selecting appropriate trust models [99]. Finally, the formal correctness and appropriateness of the proposed criteria notwithstanding, the underlying information could be vulnerable to attack. The attack models would depend on the particular protocols and data structures used to implement credential transfer and refresh. As such they are out of scope for an abstract framework.

3.5.3 Safety and Consistency of Mutable Attributes Using Quotas

To tackle concurrent usage of mutable attributes in a distributed environment we propose two categories of practical scenarios, from which we justified the distributed approach to be a better match with modern system environments and requirements. We further discuss two types of service-based and user-based subcategories. Both subcategories of distributed approach are amenable to quota-based approach. Formal specification of use cases have been proposed relying on nomenclature of UCON [150] paper. We assert that revocation check is inappropriate given mutable attributes values would be changing frequently and new values of them have to be sought and utilized in the decision making process. Therefore, refresh should be used to pull recent values of attributes from attribute authorities.

Chapter 4: USER-TO-DEVICE ADMINISTRATION OF ACCESS IN SMART HOME IOT ENVIRONMENTS

In this chapter, we propose an RBAC administrative model to govern authorization assignments for underlying operational access control model in a smart home IoT environments. Our model specifically addresses the administration of extended generalized role based access control (EGRBAC) [24] for smart home IoT. Proposed administrative model could be simply extended to manage other more sophisticated access control models with similar dynamics. We augment our proposed model by providing use case scenarios for both operational and administrative models. We propose a decentralized, ledger-based, publish-subscribe based architecture for administration of access in a smart home IoT environment to preside at the assignments of underlying operational authorizations. Proposed architecture is endorsed by a proof-of-concept implementation, which utilizes smart contracts to ensure the integrity of administration supplemented by intrinsic benefits of blockchain to be distributed and transparent. Results of this research have been published in following two peer-reviewed papers:

1. Mehrnoosh Shakarami, and Ravi Sandhu. "Role-Based Administration of Role-Based Smart Home IoT", In Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, pp. 49-58. 2021.

2. Mehrnoosh Shakarami, James Benson, and Ravi Sandhu. "Blockchain-Based Administration of Access in Smart Home IoT", In Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, pp. 57-66. 2022.

4.1 Role-Based Administration of Role-Based Smart Home IoT

In this section, we propose a role-based administrative model corresponding to the EGRBAC operational model to govern authorization functionalities, through management of important assignments in the operational model.

4.1.1 Motivation

There is a rich body of research on security of IoT [27, 29, 90, 131, 211]. Authors in [148, 157, 160, 161] review the access control requirements and approaches to protect security and privacy of IoT. Security of a smart home environment, as a specific application of IoT, has been investigated in [58, 215]. Common to all of these studies, access control has been recognized as a critical requirement to build a secure IoT environment.

Overall system's security mainly depends on both operational and administrative access control models. Administrative policies determine who can make changes to the current authorization state of the system. Administrative aspects of access control is one the most important, yet least studied areas of access control, specifically when it is required to be done in a distributed manner [120, 173]. In IoT environments, specifically when users are mostly not expected to be IT professionals, the usability of access policy administration is an important feature to be taken care of. In a smart home IoT environment, we want access management to be easy to use and minimize the required effort for house administrators (i.e. house owners/parents). For instance, access management should not impose admins to define a new policy every time a new IoT device is added to the house. It also should not be complicated, so common users with minimum IT specialty can utilize it.

Intrinsic benefits of Role-Based Access Control (RBAC) such as its policy neutrality, adherence to least privilege principle and its built-in support for Static and Dynamic Separation of Duty (SSoD and DSoD), made it a preferred choice in prior research works in order to establish an access control model for mediating access to users in a smart home, using the concept of a role [20, 24, 187, 198]. One of the improvements that RBAC made on its predecessor models is its ease of management. After an operational RBAC model has been established, the administration is facilitated by assigning different users to define roles or making changes to existing role sets of the system. However, the notion of an administrative model is not included in the NIST standard [76] nor the seminal RBAC [172] models. Administrative RBAC (ARBAC) was first proposed as an approach to use RBAC itself to manage different aspects of RBAC [170]. To date, surprisingly

little attention has been paid to design administrative models in IoT environments, despite the specific requirements for their corresponding operational access control models. In particular, the need to design administrative models arise dealing with IoT environments where users and devices are constrained.

4.1.2 An RBAC Administrative Model for Smart Home IoT

Our model specifically addresses the administration of EGRBAC [24]. However, it could be simply extended to manage other more sophisticated access control models with similar dynamics. The use of RBAC for RBAC administration enables us to separate governing of different assignments in corresponding operational model. In case of EGRBAC (as our operational model), we have different relations to be administered including assigning users to roles, defining new environmental conditions, introducing new role pairs and assignment of device roles to role pairs, each of which could be a component of administration.

We classify possible changes in smart home environment into three classes which need to be administered.

1. New User Added: A new individual could join to the set of smart home users any time, which consequently needs administrative changes to be done such as defining a new role, an environment role or a role pair. We recognize adding a new user to be an infrequent event. So, we consider this case orthogonal to central focus of this paper. Its administration would be centralized, say, in the homeowner.

2. New Device Added: Adding a new device is likely to happen increasingly frequently, considering the surge in smart home devices to be available nowadays. This change should be reflected in access control model by defining new device roles, making changes to current PDRA assignment or new assignments of permission to device roles through adding new PDRA relations. Establishment of new access control policies through managing RPDRA, is also a plausible administration requirement. In this paper, we focus on governing RPDRA and PDRA to address these requirements. We assume making changes in an existing device

role or defining a new device role is centrally managed in some way.

3. Modify Current Assignments: Sometimes it is required to change current assignments in a smart home, even if there is no change in the set of users or devices. For instance, adding a new constraint for assigning a device role to a role pair (modify RPDRA), changing the set of (device, permission) pairs which have been assigned to a device role (modify PDRA). Modifying current PDRA sometimes is required as a result of adding a new device to the system, by adding new (device, permission) pairs to current PDRA. We focus on PRDA and RPDRA administrative modifications. Although other assignment changes are plausible to be required, e.g. change a user's role (UA modification), we consider those changes out of scope.

We recognize administration to be best done if it is decentralized. Centralized administration generates a single point of failure. Moreover, even in a small environment like a smart home, decentralized administration is worthy to consider. Suppose one of the administrator users are not available to manage/delegate the access control authorizations. A decentralized approach would bring the benefit of presence of another assigned administrator user who could do the task. Decentralized administration also helps to improve user's privacy by defining all permissions to manage a user's privacy zone contained in a separate administrative unit, and specify that user as the only possible user who could be assigned to the correspondent administrative role. In this paper, decentralization has been applied on two assignment relations (PDRA and RPDRA) in EGRBAC. We develop a formal description of administrative concepts and constraints in the following.

Proposed Administrative Model in More Detail

Access control is embodied in different authorization assignments of the EGRBAC model, including UA, RPRA, PDRA, RPDRA, etc. These components collectively would establish the access control policy of the system. In this paper, we first focus on the RPDRA assignment through which device roles would be assigned to role pairs and considered as the central step of access policy establishment. Our administrative model focuses on managing the operational access control model

in a way that any legitimate user in the smart home environment only has access to what s/he is authorized to access. In other words, insider threats are limited such that our system observes the least privilege principle[1] while managing the authorization assignments.

In order to design our administrative model to be decentralized, we use the abstract of Administrative Unit (AU), which is a core component of decentralization in our model. As indicated in [172], it highly matters how to scope the administrative authority conferred to administrative roles. In our model, each administrative unit contains a unique specific Administrative Roles (AR) and a set of Administrative Tasks (AT). In other words, each administrative role is authorized to manage the administrative tasks within a given administrative unit. This authorization is scoped as a set of administrative tasks defined to manage corresponding assignments in an operational model.

The introduced concept of administrative unit in our work is comparable to the abstract of administrative scope introduced in ARH [54], which *"informally associates each role in the role hierarchy to the set of roles over which it has control"*. However, there is a twofold distinction between these concepts: first, similar to ARBAC97 [170], we assume administrative roles are separate from regular roles, while in ARH administrative roles are a set of regular roles in the system augmented with administrative authorities. Second, ARH is focused on role hierarchy administration. It considers Role-Role relation in RBAC model, in contrast to our administrative model which has a dissimilar underlying operational model and designed to manage different kinds of assignments.

We propose a basic administrative model to manage RPDRA in the operational model, and then extend it to a more generic model which is able to also manage PDRA. This extension could be generalized to construct a comprehensive administrative model which is able to manage all assignments in the operational model. We define one administrative unit per operational assignment to be managed, which includes a unique administrative role and a set of administrative tasks, as follows. The set of Administrative Tasks reflects the scope of control which is potentially available

[1] `https://us-cert.cisa.gov/bsi/articles/knowledge/principles/least-privilege`

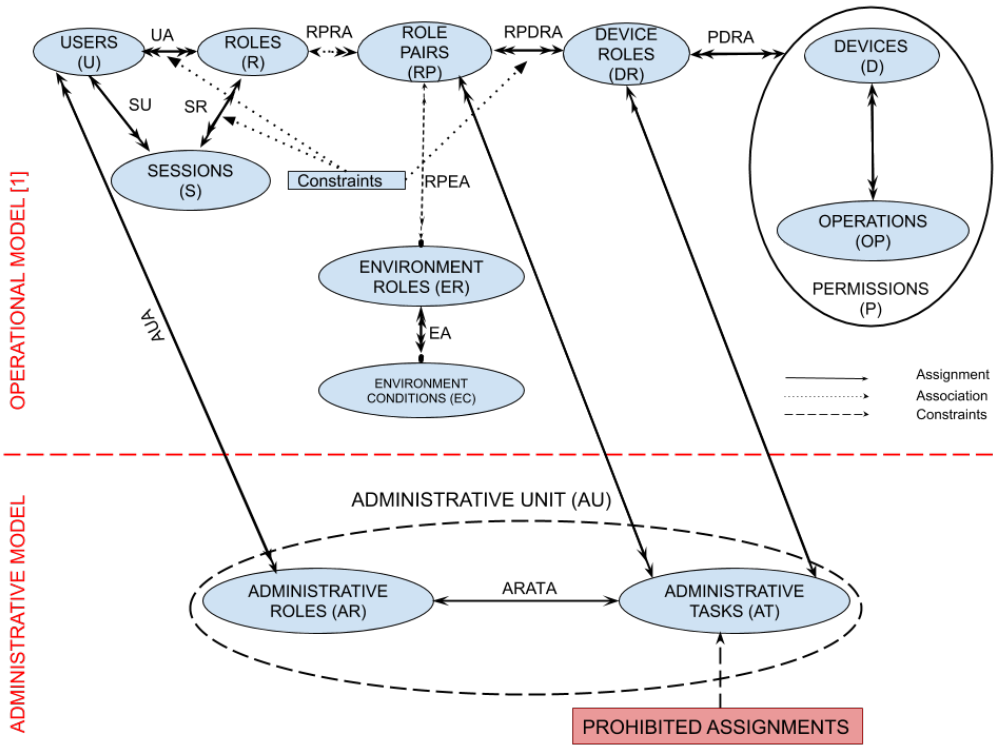

Figure 4.1: Administrative Model

to each AU's administrative roles.

RPDRA Administration In order to manage RPDRA, each Administrative Task is defined as a set which itself contains two sets: a set of Device Roles (DR), which is a subset of available device roles defined in the system and a set of Role Pairs (RP) which is a subset of available RPs in the system.

PDRA Administration For managing PDRA relation, each Administrative Task is defined as a set which includes two sets: a subset of Device Roles (DR) and a subset of permissions (P).

Formal Definition of Proposed Model

In this section, we present most notable features of our model via formalism. Formal definitions have been also presented in Table 4.1. `Core components` include the concepts of Administrator Users (AUser), Administrative Roles (AR), Administrative Unit (AU), Administrative Task

Table 4.1: Administrative Model Formalization

Core Components

$-AUser \subset U$ is a set of administrator users.

$-AR$ is a set of administrative roles, authorized to manage a specified subset of RPDRA.

$-AUA \subset AUser \times AR$ is a many to many administrator user to administrative role assignment.

$-AU$ is a set of administrative units.

$-AT \subseteq (2^{RP} \times 2^{DR}) \backslash ProhibitedAssignment$ is a set of administrative tasks, which contains all pairs of cross product of a subset of RP, and a a subset of DR, but a set of Prohibited Assignments has to be excluded.

Administrative Constraint

$-ProhibitedAssignment$ is a set of prohibited (rp, dr) pairs each of which is a member of possible pairs of assignment but specified to be forbidden by design, ($Constratints \subset RP \times DR$).

Administrative Authorization

$-ARATA \subseteq AR \times AT$ is a one to one AR to AT assignment determining the scope of administrative control for a given AR.

$-ARAUA \subseteq AR \times AU$ is a one to one AR to AU assignment, determines which AU is under control of a given AR.

Derived Administrative Relations

$-AR_{at \in AT} \subset AT \times AR : AR_{at} = ar \in AR : at \in ARATA(ar)$: many to one administrative task to administrative role function which determines which ar can manage this at.

$-RolePair_{at \in AT} \subseteq 2^{RP}$ determines which role pairs are included in a given administrative task.

$-DeviceRole_{at \in AT} \subseteq 2^{DR}$ discovers the device roles which are included in a given administrative task.

$-InclusiveTask((rp, dr)) \subseteq (rp \in RP, dr \in DR) \times \{AT \cup FALSE\}$ determines the association of a (rp, dr) to an administrative task, at, if this pair is currently defined as a member of that administrative task, if no inclusive administrative task found, it returns FALSE.

Check Access Predicate

$-$ASSIGNRPDR$(auser \in AUser, ar \in AR, rp \in RP, dr \in DR) \equiv (((auser, ar) \in AUA) \wedge (at =$ InclusiveTask(rp,dr) $\wedge\ ar = AR_{at}) \wedge ((rp, dr) \notin RPDRA)) \Rightarrow RPDRA' = RPDRA \cup (rp, dr)$

$-$REVOKERPDR$(auser \in AUser, ar \in AR, rp \in RP, dr \in DR) \equiv (((auser, ar) \in AUA) \wedge (at =$ InclusiveTask(rp,dr) $\wedge\ ar = AR_{at}) \wedge ((rp, dr) \in RPDRA)) \Rightarrow RPDRA' = RPDRA \backslash (rp, dr)$

(AT) and Administrative User Assignment (AUA).

Administrator Users (AUser) are a subset of regular users, with administrative authorizations. Administrator users would be recognized by their assignment to Administrative Roles (AR). Administrative User Assignment (AUA) is a relation which assigns administrator users to administrative roles. Administrative Unit (AU) is an abstraction to represent a unit of administration, which contains the scope of management of its contained AR. Each Administrative Unit (AU) includes two components, a uniquely associated AR and a subset of possible authorization assignments, namely Administrative Tasks (AT). Any AR included in an AU is permitted to manage any of the possible authorization assignments included in its corresponding AT. For instance, if a `Homeowner` assigned to be the AR of an AU and `scheduling the thermostat` is included in the AT included in the same AU, it implies that any user with `Homeowner` role would be authorized to manage `thermostat schedule`.

We define `Administrative Constraint` as a set of prohibited assignments which indicate denial of access instead of conferring it. That is negative permissions are modeled as constraints in our system. For instance, babysitter does not need and should not be granted access to the thermostat's schedule. `Administrative Authorizations` indicate the relation defined in order to assign of AT to AR (defining the scope of control of AR) and AR to AU (indicating the Administrator Role in an Administrative Unit). ARATA is Administrative Role to Administrative Task assignment, which is a one to one relation, which means only one AR could be authorized to activate authorizations included in corresponding AT. ARAUA is Administrative Role to Administrative Unit Assignment, which is a one-to-one relation, that means no more than one AR can be assigned to an AU. So, both AT and AU are uniquely associated to an AR. It is notable that it is always possible to assign more than one user to an AR.

`Derived Administrative Relations` are a set of functions used to retrieve administrative relations between different components of the model. These functions could be later utilized to evaluate a constraint which should be sustained in all assignments/ revocations. $AR_{at \in AT}$ indicates the AR which has control over specified at. To determine role pairs and device roles which

are included in an administrative task, functions $RolePair_{at \in AT}$ and $DeviceRole_{at \in AT}$ could be used correspondingly. $InclusiveTask((rp, dr))$ function finds out the administrative task within which the given pair of device role and role pair are included. Our model components have been depicted in Figure 4.1.

`Authorization Functions` which are represented in bottom part of Table 4.1, determining the conditions that qualify an administrator user to do assignments/revocation which completes the operational model's access policy. Proposed authorization functions decouple assignment and revocation of a specific (rp, dr), which means there is no requirement for the revoking user to be the same user who granted a specific access.

Function ASSIGNRPDR$(auser \in AUser, ar \in AR, rp \in RP, dr \in DR)$ enables a user $auser$ with ar role to add the (rp, dr) to the set of RPDRA of operational model. This means the device role *dr* would be assigned to the role pair *rp*, which consequently adds a new rule to the set of policies of EGRBAC. To qualify the requesting user, the assignment function finds the including AT of given (rp, dr) set as well as the AR which is in charge for that specific task. The model checks if the requesting administrator user has the AR which controls the retrieved AT and add the (rp, dr) to the set of RPDRA provided that the rule has not been previously created.

Similarly, function REVOKERPDR$(auser \in AUser, ar \in AR, rp \in RP, dr \in DR)$ would authorize an administrator user $auser$ with ar role to revoke a device role from a role pair by checking similar preconditions as ASSIGNRPDR, unless in case of revocation, the intended (rp, dr) should has been previously assigned by a legitimate administrator. As a result, the (rp, dr) pair would be deleted from the set of RPDRA of EGRBAC.

4.1.3 Use Case Definition

In this section we will discuss a case study of smart home in two parts of operational and administrative cases. Proposed operational use case is an extension to what has been presented in [24]. Then the corresponding administrative use case would be discussed.

Operational Use Case

Presented use case aims to make a representation of a smart home environment in which users' accesses are granted to parts of functionalities of given devices, a.k.a. device roles. Parents want children to have access only to the kids_friendly_content on entertainment devices (TV, DVD, and PlayStation). It should not be possible for kids to access to some functionalities of devices, which should be specifically controlled by an adult, for example turn the oven on/off, controlling the thermostat or garage door functionalities and so on.

Furthermore, we want babysitter to access the required adult-controlled functionalities, such as turning the oven/thermostat on/off and lock/unlock the front door. However, we do not want to grant an unnecessary access to babysitter, e.g. modifying the thermostat schedule. The most permissive users would be the parents, to whom all functionalities of smart home are available.

The operational use case can be configured as illustrated in Table 4.2. There are five users `Alex, Bob, Susan, James, Julia` who are correspondingly assigned to roles `kid, parent, babysitter, guest and parent`. The set of devices include `TV, DVD, PlayStation, DoorLock, Oven, SurveillanceCamera, BurglarAlarm, GarageDoor, Thermostat` each of which has been associated with a set of operations defined by the manufacturer.

We defined a set of permissions including 9 different permission sets. We designed the set of permissions based on available operations for each device, as well as the desired access control regulations we previously mentioned. For example, we come up with two different permissions P_8 and P_9 for thermostat. This design aims for implementing the least privilege principle, as in next steps we can assign these permission sets to different device roles, e.g., assign P_8 to `Adult_Controlled` device role. Then, we assign babysitter to this device role, it is possible to turn the thermostat on/off, but excessive access to thermostat's schedule would not be provided. Same consideration has been taken in designing separate permission sets of P_1 and P_2 for entertainment devices, so it would be possible to define a device role, `Kids_Friendly_Content`, which would provide kids with least required permissions necessary for their access. Four Device

Table 4.2: Operational Use Case

U = {Alex, Bob, Susan, James, Julia}
R = {kid, parent, babySitter, guest}
UA = {(Alex,kid), (Bob,parent), (Susan,babySitter), (James,guest), (Julia,parent)}
D = {TV, DVD, PlayStation, DoorLock, Oven, SurveillanceCamera, BurglarAlarm, GarageDoor, Thermostat}
OP = {On, Off, PG, R, Lock, Unlock, Activate, Deactivate, On_{Oven}, Off_{Oven}, StartRecording, StopRecording, $\text{Open}_{\text{GarageDoor}}$, $\text{Close}_{\text{GarageDoor}}$, $\text{On}_{\text{Thermostat}}$, $\text{Off}_{\text{Thermostat}}$, $\text{Schedule}_{\text{Thermostat}}$}
$P_1 = \{TV, DVD, PlayStation\} \times \{On, Off, PG, R\}$
$P_2 = \{TV, DVD, PlayStation\} \times \{On, Off, PG\}$
$P_3 = \{Oven\} \times \{On\_Oven, Off\_Oven\}$
$P_4 = \{FrontDoor\} \times \{Lock, Unlock\}$
$P_5 = \{SurveillanceCamera\} \times \{StartRecording, StopRecording\}$
$P_6 = \{BurglarAlarm\} \times \{Activate, Deactivate\}$
$P_7 = \{GarageDoor\} \times \{Open_{GarageDoor}, Close\_GarageDoor\}$
$P_8 = \{Thermostat\} \times \{On\_Thermostat, Off\_Thermostat,$
Schedule_Thermostat}
$P_9 = \{Thermostat\} \times \{On\_Thermostat, Off\_Thermostat\}$
$P_{10} = \{OutdoorCamera\} \times \{On_{OutdoorCamera}, Off_{OutdoorCamera}\}$
$P = \bigcup_{i=1..10} P_i$
DR = {Entertainment_Devices, Adult_Controlled, Owner_Controlled, Kids_Friendly_Content }
PDRA = $\{P_1 \times Entertainment\_Devices\} \cup \{P_2 \times Kids\_Friendly\_Content\} \cup \{\{P_3 \cup P_4 \cup P_9\} \times Adult\_Controlled\} \cup \{\{P_5 \cup P_6 \cup P_7 \cup P_8\} \times Owner\_Controlled\}$
EC = {weekends, evenings, vacation, TRUE}
ER = {Entertainment_Time, Any_Time, Not_At_Home}
EA = {({weekends, evenings},Entertainment_Time), ({vacation},Not_At_Home), ({TRUE},Any_Time)}
RP = {(kid,{Entertainment_Time}),(parent,{Any_Time}),
(babySitter,{Any_Time}),(guest,{Any_Time}), (parent,{Not_At_Home})}
RPDRA = {((parent,{Any_Time}),Adult_Controlled),
((parent,{Any_Time}),Owner_Controlled),
((parent,{Any_Time}),Entertainment_Devices),
((kid,{Entertainment_Time}),Kids_Friendly_Content),
((babysitter,{Any_Time}),Adult_Controlled),
((guest,{Any_Time}),Entertainment_Devices)}

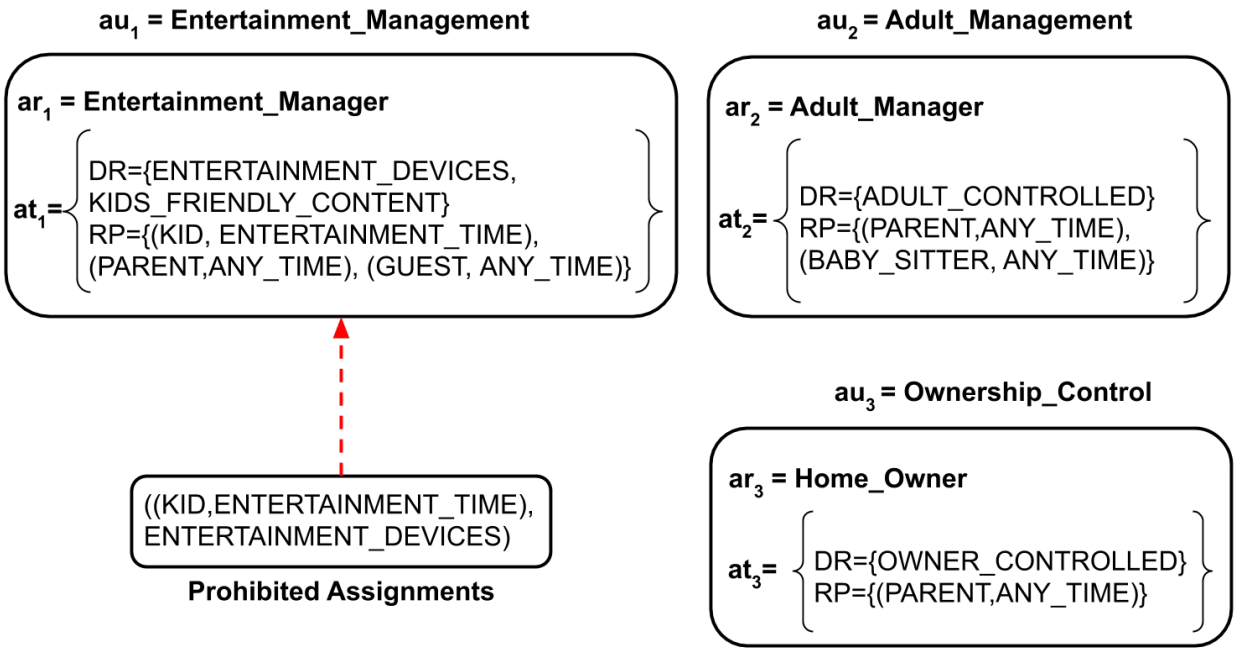

Figure 4.2: Administrative Units

Roles have been introduced and different permission sets have been assigned to them using PDRA.

A set of Environment Conditions, EC, has been assigned to different Environment Roles, ER, which would be later coupled by Roles to create Role Pairs, RP. Coupling device roles with role pairs through RPDRA completes the set of access rules in the system. As an instance, the ((parent,Any_Time),Adult_Controlled) pair communicates that parent can access to Adult_Controlled device role, which includes access to turn the oven and thermostat on/off and lock/unlock the front door, at any time.

Administrative Use Case

Table 4.3 depicts the administrative use case based on our proposed model and corresponds to the operational use case discussed in previous section. Administrator users are a subset of regular users in the operational use case, and include Bob and Julia. As illustrated in Table 4.3, each of administrator users has been assigned to two different administrative roles. Bob has been assigned to Home_Owner and Entertainment_Manager and Julia has been assigned to Home_Owner and Adult_Manager. Note that the Home_Owner administrative role has both

Table 4.3: Administrative Use Case

AUser = {Bob, Julia}
AR = {Entertainment_ Manager, Home_Owner, Adult_Manager}
AUA = {(Bob, Home_Owner), (Julia, Home_Owner), (Julia, Adult_Manager), (Bob, Entertainment_ Manager)}
AU = {Entertainment_Management, Ownership_Control, Adult_Management}
ProhibitedAssignment = {((kid,{Entertainment_Time}), Entertainment_Devices)}
AT = {at$_1$, at_2, at_3}
at$_1$ = $\{(parent, \{Any\_Time\}), (babysitter, \{Any\_Time\})\} \times \{Entertainment\_Devices, Kids\_Friendly\_Content\} \setminus$ {ProhibitedAssignment}
at$_2$ = $\{(parent, \{Any\_Time\}), (babysitter, \{Any\_Time\})\} \times \{Adult\_Controlled\} \setminus \{ProhibitedAssignment\}$
at$_3$ = $\{(parent, \{Any\_Time\})\} \times$ {Owner_Controlled} $\setminus \{ProhibitedAssignment\}$
RolePair(at$_1$) = $\{(parent, \{Any\_Time\}), (guest, \{Any_Time\}),$ (kid,{Entertainment_Time})}
DeviceRole(at$_2$) = $\{Adult\_Controlled\}$
InclusiveTask((kid,{Entertainment_Time}), Kids_Friendly_Content) = at$_1$
ARATA = {(Entertainment_Manager,at_1),(Adult_Manager,at_2), (Home_Owner,at_3)}
$AR_{at_1} = \{Entertainment\_Manager\}$
$AR_{at_2} = \{Adult\_Manager\}$
$AR_{at_3} = \{Home\_Owner\}$
assignRPDR(Bob, Entertainment_Manager, ({(kid,{Entertainment_Time}), Kids_Friendly_Content})) $\Longrightarrow$ RPDRA = RPDRA $\cup$ {((kid,{Entertainment_Time}), Kids_Friendly_Content)}
revokeRPDR(Bob, Entertainment_Manager, ({(kid,{Entertainment_Time}), Kids_Friendly_Content})) $\Longrightarrow$ RPDRA = RPDRA $\setminus$ {((kid,{Entertainment_Time}), Kids_Friendly_Content)} $\Longrightarrow RPDRA = \emptyset$
assignPDR(Julia, Home_Owner, P$_{10}$, $Owner\_Controlled) \Longrightarrow$ PDRA = PDRA $\cup$ {(P$_{10}$, $Owner\_Controlled)\} revokePDR(Julia, Home\_Owner, P_3, Adult\_Controlled) \Longrightarrow$ PDRA = PDRA $\setminus$ {(P$_3$, $Adult\_Controlled)\}$

Bob and Julia as administrator, which addresses the single point of failure associated with centralized administration.

There are three Administrative Units (AU) defined in our smart home use case including Entertainment_Management, Ownership_Control and Adult_Management. There is one AR uniquely associated with each AU, so an admin unit cannot have more than one AR in charge of it, but more than one administrator user could be assigned to that AR. There is one Administrative Task (AT) in each AU, which includes a set of DR and a set of RP. AU can grant any subset of $RP \times DR$ using the ASSIGNRPDR authorization functions or revoke by using REVOKERPDR.

There are some samples of administrative relations depicted in Table 4.3. For instance, AR_{at_1} indicates the AR which is in charge of at_1. Authorization functions have also been shown, for example ASSIGNRPDR = (Bob, Entertainment_Manager, ({(kid,{Entertainment_Time}), Kids_Friendly_Content})), would first find the inclusive task of the given (rp,dr) which is at_1. It then checks if Entertainment Manager is the AR assigned to ({(kid,{Entertainment_Time}), Kids_Friendly_Content}), which is true in this use case. Lastly, if the requested access pair is not previously defined, it would add it to the access rules in RPDRA of the operational model.

It is noteworthy to see the ((kid,{Entertainment_Time}), Entertainment_Devices) has been defined as ProhibitedAssignment, so the authorization function (ASSIGNRPDR) would not let any administrator to grant access to Entertainment_Devices to the kids at their entertainment time. The illustration of discussed usecase has been provided in Figure 4.2.

4.1.4 Administrative Model Extension

In this section, we extend our model to support PDRA assignments as well. So, when a new device added to the smart home environment, its permissions could be assigned to an existing/newly created device role/s by adding PDRA assignments. Also, an administrator might decide to rearrange

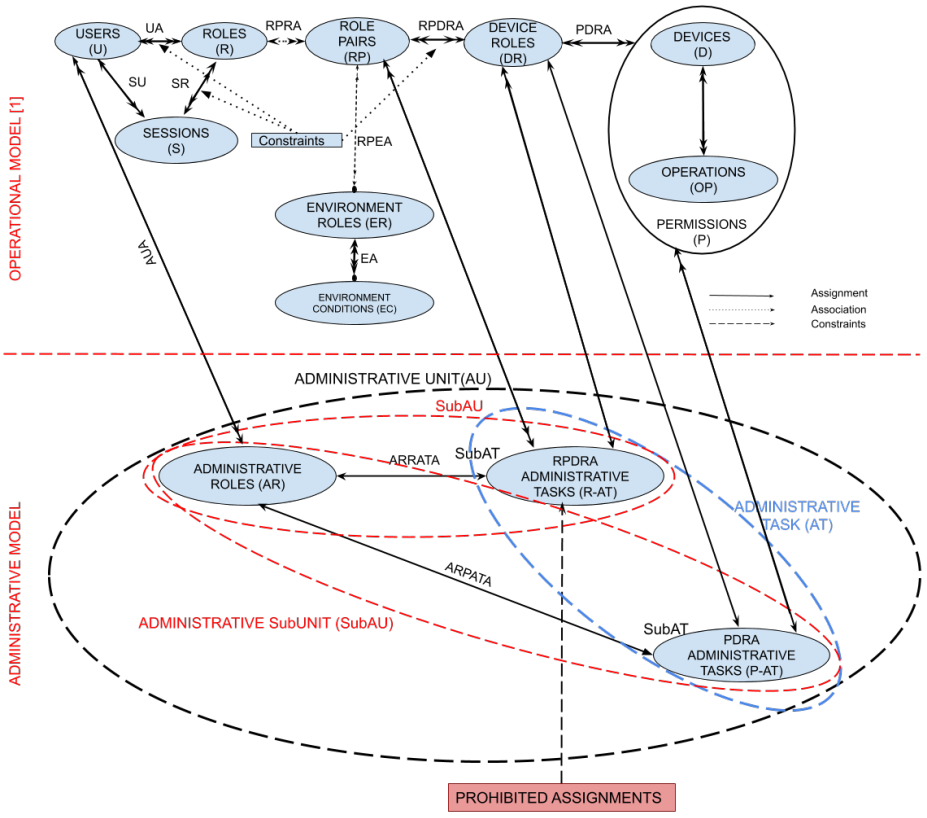

Figure 4.3: Extended Administrative Model for Managing both RPDRA and PDRA

permissions associated to a device role, which is accomplished through making changes to PDRA. Succinctly, we extended previous administrative model by defining different administrative sub-units, each of which includes an AR and an administrative sub-task. Consequent changes to the model formalization has been proposed, that we will review in this section. Proposed extended administrative model is illustrated in Figure 4.3.

Formal Definition of Extended Model

The formal definition of previous administrative model has been extended as depicted in Table 4.4. We adopt the same administrative functional categories as our first model. The same concept of Administrative Unit (AU) exists in the extended model. Here, AU would encompass two sub-units (SubAU), one for governing PDRA and another for managing RPDRA. Similarly, the set of Administrative Tasks (AT) includes two Administrative SubTask (SubAT), naming P-AT which

includes the subAT which corresponds to PDRA and another subAT named R-AT which contains ATs correspond to RPDRA.

R-At is the same as AT in previous version of our model and contains two sets, a subset of role pairs (RP) and a subset of device roles (DR). P-AT includes two sets, one is the subset of permissions (P) and another is a subset of device roles (DR). There is a unique set of Administrative Roles (AR). Each Sub-AU includes a SubAT and an AR which has been uniquely assigned to manage that SubAT. This Assignment would be a one-to-one relationship, however it is possible for one AR to be administrator for different SubATs.

Corresponding administrative authorizations have been added to the model formalism, as AR-RATA is a one to one relationship which uniquely assigns an administrative role (AR) to a R-AT administrative subtask. Likewise, ARPATA assigns a unique administrative role (AR) to a P-AT administrative subtask via a one to one relation.

Authorization functions which control over manipulating RPDRA remain the same. We also added analogous authorization functions for PDRA management, which have been shown at the bottom of Table 4.4.

Function ASSIGNPDR$(auser \in AUser, ar \in AR, p \in P, dr \in DR)$ enables a user $auser$ with ar role to add the (p, dr) to the set of PDRA of the operational model. As an illustration, suppose a smart outdoor camera has been added to smart home, with the permissions set to be $\{OutdoorCamera\} \times \{On, Off\}$. So, an authorized administrator user, any of homeowners, should be able to assign this new permission to previously/newly defined device roles. In this example, the new permission could be assigned to $Owner\_Controlled$ device role in the system. Required changes to represent this usecase have been color-coded in Tables 4.2 and 4.3.

Equivalently, required changes in Function REVOKEPDR$(auser \in AUser, ar \in AR, p \in P, dr \in DR)$ enables the user $auser$ with ar role to remove the (p, dr) from the set of PDRA of operational model. So, any role pair which has been coupled with device role dr would consequently lose the permission p. For instance, as depicted in Table 4.2, Julia as the an administrative user with the administrative role Home_Owner can remove the permission $P_3 = \{Oven_{On}, Oven_{Off}\}$ from

Table 4.4: Extended Administrative Model Formalization

Core Components

$-AUser \subset U$ is a set of administrator users.

$-AR$ is a set of administrative roles, authorized to manage a specified subset of RPDRA.

$-AUA \subset AUser \times AR$ is a many to many administrator user to administrative role assignment.

$-AU = \cup_{\forall i} SubAU_i$,is a set of administrative sub-units (SubAU).

$-AT$ is a set of administrative sub-tasks (SubAT), i.e. $AT = P\text{-}AT \cup R\text{-}AT$.

$-R\text{-}AT \subseteq (2^{RP} \times 2^{DR}) \backslash ProhibitedAssignment$ is a set of administrative tasks related to RPDRA assignment, which contains all pairs of cross product of a subset of RP, and a a subset of DR, but a set of ProhibitedAssignments has to be excluded.

$-P\text{-}AT \subseteq (2^{P} \times 2^{DR})$ is a set of administrative tasks related to PDRA assignment, which defines permission assignments to device roles.

$-SubAU \subset AR \times \{R\text{-}AT, P\text{-}AT\}$ is a administrative sub-unit.

Administrative Constraint

$-ProhibitedAssignment$ is a set of prohibited (rp, dr) pairs each of which is a member of possible pairs of assignment but specified to be forbidden to be added to RPDRA by design, ($ProhibitedAssignment \subset RP \times DR$).

Administrative Authorization

$-ARRATA \subseteq AR \times R\text{-}AT$, is a one to one AR to R-AT assignment determining the scope of administrative control for a given *ar* on RPDRA.

$-ARPATA \subseteq AR \times P\text{-}AT$, is a one to one AR to P-AT assignment determining the scope of administrative control for a given *ar* on PDRA.

$-ARAUA \subseteq AR \times AU$ is a one to one AR to AU assignment, determines which au is under control of a given ar.

Derived Administrative Relations

$-AR_{at \in SubAT} \subset SubAT \in AT \times AR : AR_{at} = ar \in AR : at \in ARRATA(ar) \vee at \in ARPATA(ar)$: many to one administrative subtask to administrative role function which determines which *ar* can manage this *at*.

$-RolePair_{at \in AT} \subseteq 2^{RP}$ determines which role pairs are included in a given administrative task.

$-DeviceRole_{at \in AT} \subseteq 2^{DR}$ discovers the device roles which are included in a given administrative task.

$-InclusiveTask((rpp, dr)) \subseteq (\{(rpp \in RP) \vee (rpp \in P)\}, dr \in DR) \times \{AT \cup FALSE\}$ determines the association of a (st, dr) to an administrative task (either R-AT or P-AT) if this pair is currently defined as a member of that administrative task, if no inclusive administrative task found, it returns FALSE.

Check Access Predicate

$-$ASSIGNRPDR$(auser \in AUser, ar \in AR, rp \in RP, dr \in DR) \equiv (((auser, ar) \in AUA) \wedge (r\text{-}at =$ InclusiveTask(rp,dr) $\wedge\ ar = AR_{r-at}) \wedge ((rp, dr) \notin RPDRA)) \Rightarrow RPDRA' = RPDRA \cup (rp, dr)$

$-$REVOKERPDR$(auser \in AUser, ar \in AR, rp \in RP, dr \in DR) \equiv (((auser, ar) \in AUA) \wedge (r\text{-}at =$ InclusiveTask(rp,dr) $\wedge\ ar = AR_{r-at}) \wedge ((rp, dr) \in RPDRA)) \Rightarrow RPDRA' = RPDRA \backslash (rp, dr)$

$-$ASSIGNPDR$(auser \in AUser, ar \in AR, p \in P, dr \in DR) \equiv (((auser, ar) \in AUA) \wedge (p\text{-}at =$ InclusiveTask(p,dr) $\wedge\ ar = AR_{p-at}) \wedge ((p, dr) \notin PDRA)) \Rightarrow PDRA' = PDRA \cup (p, dr)$

$-$REVOKERPDR$(auser \in AUser, ar \in AR, p \in P, dr \in DR) \equiv (((auser, ar) \in AUA) \wedge (p\text{-}at =$ InclusiveTask(p,dr) $\wedge\ ar = AR_{p-at}) \wedge ((p, dr) \in PDRA)) \Rightarrow PDRA' = PDRA \backslash (p, dr)$

the set of permissions of *Adult_Controlled* device role. So, any user with that role, e.g., babysitter, would no longer has the permission to turn the oven on/off. She might want to add that permission later to another device role, e.g. *Owner_Controlled*. This example has been added in red to the bottom of Table 4.3.

4.2 Blockchain-Based Administration of Access in Smart Home IoT

There are numerous access control models proposed for IoT environments, however, many of the proposals have remained at the conceptual level. Even so, different deployment mechanisms in some frameworks relying on existing technologies, including cloud [39, 66], Open Authorization (OAuth) [177] and blockchain [123, 141, 146, 147, 155], among them blockchain has been widely used in recognition of its transparency which benefits auditability. Moreover, blockchain's distributed nature removes the need to trust the third parties, which is of advantage to privacy protection. Nonetheless, using blockchain for access control is still controversial [98]. The performance of blockchain-based systems is still not competitive with current centralized access control systems. In time-sensitive applications using blockchain for access control would negatively affect users' experience [126, 168]. On the other hand, communication with blockchain demands higher amounts of computation power and space that is available to resource/energy constrained IoT devices.

In this section, we propose using blockchain for administration of access, unlike most of previously presented approaches which have been applied in operational environments. We recognize blockchain could bring its intrinsic advantages of distribution, transparency, and scalability to the administration of access while it is not yet practical to be used for operational access control. Using blockchain for enforcing the administrative model would equip the access management with improved security and posteriori auditing [174] as discussed in Section 4.2.1, as well as the potential of generalization of the proposed approach to environments with similar dynamics relying on scalable nature of the blockchain. Since administrative access control tasks are less frequent, blockchain's monetary costs and time burdens are bearable, for instance it is reasonable to wait for

seconds for an administrative change to take effect.

We discuss our enforcement architecture based on blockchain for access administration in the smart home IoT, while Greengrass [4] has been utilized to mediate device-cloud connections. It also handles required access control tasks in the local environment. Greengrass in the corresponding operational model serves as the smart hub and policy engine. So, the blockchain burdens of time, computational power and storage would not be imposed at the operational level. There is no need to store the ledger information or even communication wallets on resource-constrained IoT devices, as we do not use blockchain at the operational level for access control.

To build our enforcement architecture, we adopt the administrative model presented in previous section which is an RBAC model for administration of access in the smart home IoT environment. This model provides decentralization, scalability, and generalization via defining administrative units to scope the administrative tasks entitled to each administrator user, which also benefits users' privacy. EGRBAC (Extended Generalized RBAC) [24] is picked in [186] as the underlying operational model which is a dynamic, fine-grained, and context-aware operational model. Our proposed architecture provides interoperability of administrative and operational levels of access. Besides proposing an enforcement architecture, different interactions to it are presented in a sequence diagram and it is also backed up by a proof-of-concept implementation.

4.2.1 Problem Statement and Motivation

Security of any management system is of utmost importance, so would be administration of access in a smart home. The proposed architecture in this paper is intended for administration of access for user to device interactions. It relies on blockchain's intrinsic characteristics of being immutable, tamper resistant and transparent for security provision.

Threat Model. In our proposed architecture IoT devices are not part of the blockchain network for obvious performance benefits elaborated in the next section (See 4.2.2). Thus, IoT devices would rely on access authorization rules made by access administrators. A malicious insider or an attacker could target the smart home's security by spoofing (impersonating as access manager),

tampering (modifying the access control policy towards his/her desired intent), privilege escalation (trying to elevate the available privileges or repudiation (denial of performing an action).

Motivating Example. A simple example of an insider security threat could be a dishonest babysitter trying to tamper with access control rules so that s/he would have access to the house at the times s/he is not meant to. As another example an attacker can fake an administrator account enforcing an IoT device to maliciously deny the access to a subject even though the policy would have granted it.

How Blockchain Helps with Security. In our proposed architecture administrator account and access management tasks cannot be forged or manipulated, as we utilize a wallet private key to encrypt an administrator account and definition/configuration of access, so the administrator account cannot be faked. The access administration policy is defined by programming a smart contract and is recorded into the ledger via a consensus process which is protected from tampering relying on the irreversible nature of blockchain. Therefore, it is impossible to change the authorization rules in favor of a malicious insider or an attacker, as stated in the above example. Moreover, access administrative requests are submitted via transactions which helps to verify proper implementation and integrity of access control rules. In our system, the operational access control policy is updated accordingly with transaction logs of the blockchain that ensures authorization rules' authenticity.

Furthermore, a blockchain based solution equips the system with transparency and auditability. So, if any unduly granting/denial of access happens, intrinsically immutable logs of transactions on blockchain provides a way for posteriori auditing and verifying the related policy on the chain. In case of maliciously denying access, using blockchain would equip the system with means to verify which policy was enforced, and if the policy is disobeyed by an IoT device, it reveals the device being maliciously controlled.

However, we are assuming users' communications with the edge services are secured over the home local network. Routing attacks which could stop Greengrass from receiving updates from the cloud, and attacks against web3 API compromising credentials are considered out-of-scope of

Figure 4.4: Whether a Blockchain is the Appropriate Technical Solution for Your Problem [205]

this paper. Security considerations are discussed in further details in Section 4.3.

4.2.2 Blockchain For Access Control

In this section we discuss the usage of blockchain for administration of access.

Why Not to Use Blockchain at Operational Level

As one of the application domains of blockchain, access control in the IoT domain gained a lot of attention in the literature. There are a handful of publications which recognize blockchain-based access control would strengthen overall IoT security [64, 113, 165], while others assert blockchains are not yet ready for mass usage in any domain, for which their designs and code bases have to be more mature [61]. Many of the previously proposed researches use blockchain at the operational level for access control, as briefly discussed in Section 2.2.2. Nonetheless, there are some inherent characteristics of blockchain which make it unsuitable for that purpose.

Elaborating on different approaches to use blockchain for operational access control, consider the following options: In the case of device democracy, which has been advocated by IBM as the future of IoT [3], each IoT device takes responsibility for its own access control. However, not every IoT device could be burdened with required storage and computational power, as many IoT devices are currently energy- and resource-constrained (e.g., light sensors or wearable IoT devices).

Other approaches for blockchain-based access control, use the blockchain as the storage for access control policies [145, 146]. So, every time a policy appended to the set of access control policies or retrieved from the blockchain a transaction should be communicated and confirmed. The required duration of confirming a transaction is inappropriate for operational access control purposes in which a user cannot wait ten minutes for a transaction to be completed [146]. More-

over, some actions might be latency-sensitive, for example when a wearable health IoT device should make an emergency call to 911. As one of the most popular blockchains utilized in access control, Ethereum has the average block time (the time it takes for a block to be added to the blockchain) of 13 seconds [9], which is still significant for a home user to get access to the door lock, for example. In recent research [195], authors implemented their operational access control approach based on an alliance chain built on Ethereum, yet the access control time is in the order of seconds and varies based on number of access requests.

Another problem of using blockchain for operational access control is financial, as every transaction needs a fee to be paid in cryptocurrency to be completed. Considering how recurrent the access transactions would be even in a small IoT environment like a smart home, the monetary burden could be prohibitive. The fluctuating price of cryptocurrency aggravates this problem.

Blockchain for Administration of Access

Authors in [205] presented a flowchart which shows their standing about the necessity of using blockchain for different use cases. We followed the proposed chart to justify using blockchain in this paper for administration of access in the smart home IoT environment which has been depicted in Figure 4.4, and indicates our position with utilization of blockchain for administration of access.

Blockchain's features of distributed nature, scalability and transparency make it an appealing infrastructure for access control implementation. Moreover, it could equip the system with essential security benefits, which otherwise cannot be provided using common centralized approaches as explained in Section 4.2.1. In this paper we suggest utilizing blockchain for *administrative* access control, not at the operational level, for following reasons:

- Administrative access control tasks are infrequent compared to required operational access authorizations [57], so the burden of required processing time for blockchain adoption is few and far between and worth its benefits.

- As blockchain is an immutable ledger, it provides accountability for administrative tasks. So, access control would be coupled with auditing as a posteriori analysis [174], providing

a more complete security solution.

- As the adopted administrative model in this paper is decentralized in nature, it could take benefits of blockchain decentralization to be scaled. So, proposed enforcement architecture could be extended to environments with similar dynamics, e.g., smart buildings. Moreover, relying on the distributed nature of blockchain, we can get around privacy concerns which arise when using third parties in other infrastructure, e.g., cloud.

- The need for storing blockchain information or being involved in heavy computations would be eliminated for resource constrained IoT devices, as those would not be engaged in administration of access.

A distinct feature of our research is to follow the PEI model (Policy, Enforcement Architecture and Implementation) as our reference model [169], which would be further described in Section 4.2.4. Briefly saying, we rely on an RBAC as the policy model (P in PEI) designed for administration of smart home environments [186]. Almost all the previous works, nevertheless, lack the support of a formal model and rely on informally assumed policy objectives to build their access control frameworks. We then propose our enforcement architecture (E in PEI), implemented (I in PEI) on the Ropsten testnet of Ethereum. Our research is one of the very few works [141] in which administration of access has been considered.

4.2.3 PEI: Underlying Administrative Policy

Before discussing our blockchain-based architecture for administration, we briefly describe the RBAC administrative model proposed in [186] which is built upon EGRBAC [24] as underlying operational access control model. There have been multiple studies conducted recently to understand the needs and preferences of smart home users. These studies reported smart home users expressed the need for a fine-grained access control system, and RBAC was reportedly the most preferred approach by users for limiting the access to smart home resources [93, 213, 214]. Adopting EGRBAC as the operational model provides a fine-grained RBAC access control which

provides on permission level, instead of device level access provision; so, it would be possible to grant partial access to a device by defining the device role (DR) instead of the whole device control. For instance, a babysitter can turn on/off the AC but is not permitted to change its schedule. EGRBAC captures the environmental context by defining the environment roles (ER) which later would be paired by standard user roles to create the role pairs (RP). RP and DR would later be coupled together to establish the access authorization rules.

EGRBAC is chosen not as a de-facto operational model, but because it has the desired properties for a smart home IoT operational access control on one hand and its enforcement architecture relies on AWS Greengrass [4] which can be best integrated with our enforcement of corresponding administrative model [186]. However, the proposed administrative model in [186] and hence the proposed enforcement architecture in this paper could be utilized for any underlying operational model, regardless of if it being RBAC or used any other access control paradigm.

Administrative Model

Access administration in a smart home environment is a particular access management problem as home users lack the expertise of a typical system administrator and are unlikely to spend much time learning complex interfaces to assign/revoke access rights or auditing the access logs. The other complication stems in multiple ownership for smart devices in the home which demands for decentralized access management. Moreover, to avoid a single point of failure it is required to have multiple administrators in the house. For example, if one of the home administrators is on a business trip and there is a problem to the house power system, there should be another administrator who can grant access to the electrician to fix the issues [109].

We adopt a role-based administrative model [186] which corresponds to EGRBAC operational model and governs the authorization functionalities in a smart home in a decentralized way. The decentralization is provided through defining the administrative units (AU), each of which is controlled by an administrative role (AR) which could be taken by multiple administrator users. Each administrative unit controls a predefined set of administrative tasks (AT) which represents the scope

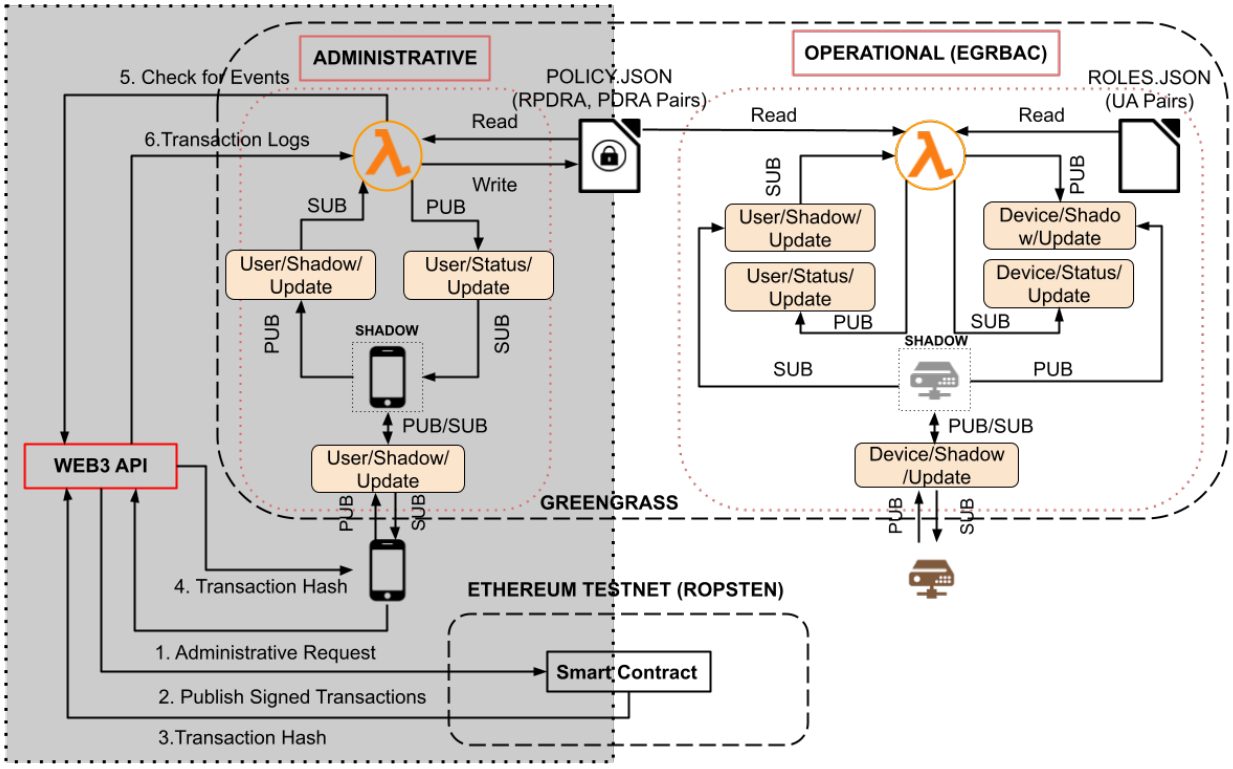

Figure 4.5: Blockchain-Based Enforcement Architecture for Administration of Access in IoT Smart Home Environment

of administration. Adopted administrative model classifies possible changes in a smart home IoT environment to be add/remove a user, add/remove a device and modifying the current operational assignments, among which adding a new user is done infrequently and could be done in a centralized way; so, is out of the scope. Therefore, the administrative tasks have been defined as management of the assignment relations in the underlying operational model. Although the model is defined in the smart home context, it could be applied to environments with similar dynamics by defining extra administrative units.

4.2.4 PEI: Enforcement Architecture

Blockchain-Based Enforcement Architecture

In this chapter, we consider the access control framework to be based on a three-layer PEI as coined in [169]. PEI stands for Policy (Policy Models), Enforcement and Implementation. Policy layer

is specified based on any access control paradigm in an ideal context which assumes all relevant information for making access decisions are instantly and securely available. The Enforcement layer manifests the policy model and provides an enforcement architecture which approximates a correspondent of the policy. Implementation layer deals with detailed implementation technologies and mechanisms.

In this paper, the policy model is adopted from [186] which is a RBAC administrative model. The enforcement architecture in this paper is compatible with the Access Control Oriented (ACO) architecture for cloud-enabled AWS IoT [23, 39], which is enclosed in the gray square with dotted border in Figure 5.5. Our authorization solution is deployed utilizing the AWS Greengrass SDK [4], an edge run-time and cloud service which provides local messaging, processing and data management services. We designed our administrative enforcement so as to be interoperable with its underlying operational model [24] which has been shown at the right side of Figure 5.5. As depicted, the POLICY.JSON file is shared between administrative and operational models; so we chose to protect its integrity with locks against possible concurrent accesses.

Overall, we propose a system which leverages the tight coupling between our authorization design and a publish/subscribe syndication which is specifically useful in the context of smart home IoT environment. Followings are the main components of the Greengrass part of the architecture which runs locally and serves as the smart hub and policy engine in our access management framework:

- Virtual objects (shadows) serve as intermediaries between applications and physical devices and keep the latest known state of the corresponding device. So, the device's state would be available to applications and services even if the device itself is not connected to AWS.

- AWS utilizes a policy-based authorization mechanism. The policies are contained in a JSON file, which includes access control rules for utilizing a resource.

- Lambda functions (λ) are event-driven computational units, sitting and waiting for messages from the topics to which they have been subscribed. As a message is received, the

lambda function wakes, does the computation reaching out to POLICY.JSON file, and publishes the results to subscribed topics. We have adopted the *operational* lambda from [24] which executes the operational level policies for user-to-device access requests. We define an *administrative* lambda, which updates the POLICY.JSON file. This file is spelled out based on administrative requests submitted by administrator users, which then would be evaluated based on the administrative policy encoded into smart contracts on the blockchain. Since the policy file is shared between both operational and administrative lambda functions, its integrity should be protected during concurrent accesses by using locks or other concurrency control mechanisms.

- Communications are done through MQTT protocol, which is a lightweight machine-to-machine publish/subscribe messaging protocol, designed for constrained devices. Local MQTT publish/subscribe messaging defines the subscriptions between publishers and subscribers.

As depicted on the left side of the Figure 5.5, we utilize Ethereum blockchain. Ethereum is a decentralized, public, permissonless, and the most actively used blockchain based on Bloomberg [10]. It is the maturest blockchain in terms of code base, user base and developer community. Ethereum is capable of being configured as both a permission-less and a permissioned blockchain network, as well as the community-based development of the platform. In other words, saying Ethereum is a permissionless blockchain means there is no authority on a network level. The logic deployed on the chain, in the form of a smart contract, does define permissions. In a smart contract, we can define an action that may only be performed by the contract's owner and not by the others.

The whole architecture represented in Figure 5.5 depicts a scenario of an administrator user at home, in which user can communicate with the Greengrass using the local home network on his/her smartphone. If the user is out of home, however, the administrator user's smartphone would have to communicate with its shadow on the AWS cloud and update it by sending a HTTP request to the AWS IoT Core. Then, the cloud forwards user's request messages to the local Greengrass by

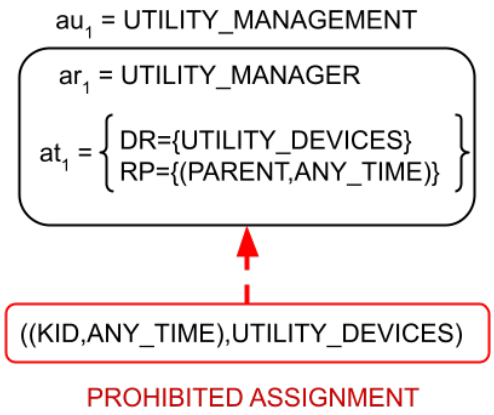

Figure 4.6: Reference Example

publishing to the user's private topic of USER/SHADOW/UPDATE, which afterwards followed by the same steps as illustrated in Figure 5.5. At the end, the user's phone shadow on the AWS cloud would update the user's phone with the status of the submitted request.

4.2.5 Sequence Diagram

For an administrative access request to be handled, a workflow as depicted in the reference scenario in Figure 4.7 is followed. An administrative user who wants to define/change the assignments of the operational model first submits the request through his/her smartphone. Consider the following example scenario depicted in Figure 4.6 in which Bob is the parent and the administrator of the smart home. There is an administrative unit (AU) called UTILITY_MANAGEMENT with the UTILITY_MANAGER as its administrative role (AR) which has been assigned to Bob. So, Bob is the administrator user who can decide about accesses of different Role Pair (RP) and Device Roles (DR) which are included in the corresponding administrative task (AT). The Device Role, UTILITY_DEVICES, includes the permissions of TURNON, TURNOFF, RESET, SCHEDULE for devices AC,FUSEBOX,WATERMETER. If Bob, as the UTILITY_ADMINISTRATOR, wants to grant the available permissions to a technician for a period of time, he should define the assignment the role pair (RP) (TECHNICIAN,REPAIR_TIME) to the UTILITY_MANAGEMENT device role. It is noteworthy that assigning kids at any time to the UTILITY_DEVICES has been defined as prohibited (refer to Figure 4.6).

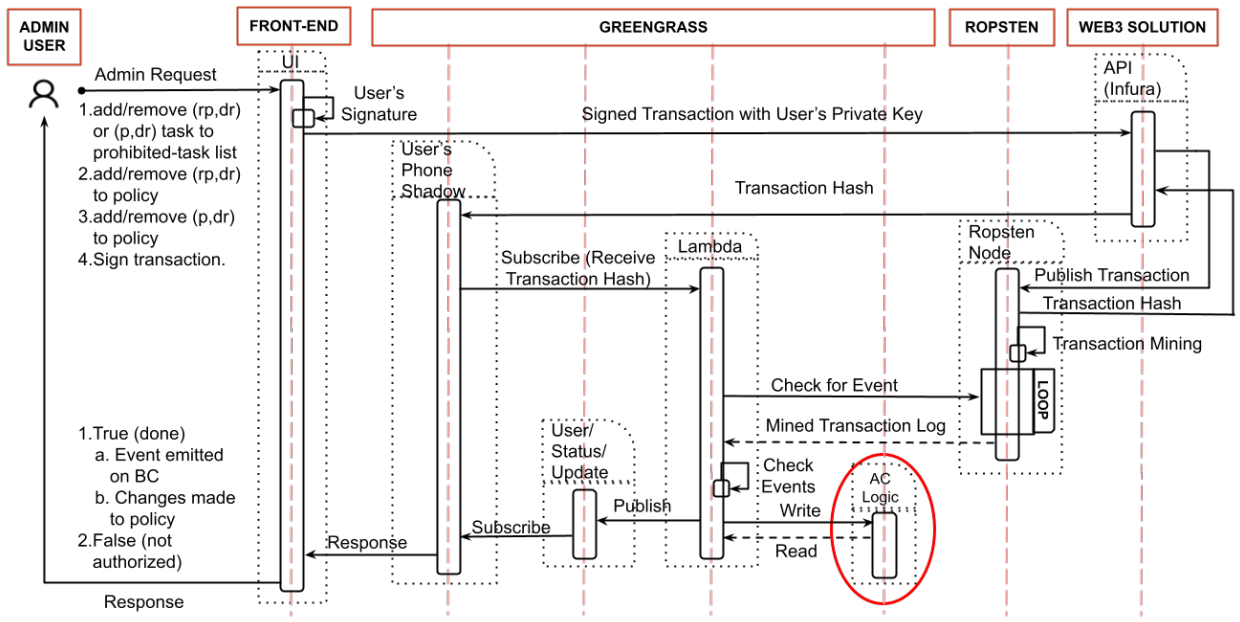

Figure 4.7: Time-Based Flow of Administration of Access Based On Proposed Architecture

For the above-mentioned example to go through, Bob must determine the desired RP and DR, sign the administrative assignment with his private key (which is securely stored on his personal smartphone) and then submit this administrative request as a transaction to the blockchain. Communication with the blockchain is conducted through a web3 API via HTTP requests. We used Infura [13] as our Web3 API in this paper. After publishing the transaction to the blockchain and getting back the transaction hash, this hash would be returned to the user's phone and also immediately published to the USER/SHADOW/UPDATE. As λ function has been subscribed to the same topic, it would also be notified with the transaction hash, which would be later used to retrieve the transaction log after being mined by λ. Administrative λ would investigate the transaction log in order to find out if the request has been approved. In either scenario of approve/deny (correspondingly permit/deny), the administrative λ would publish the results to the USER/STATUS/UPDATE topic, so the user would know the results. If permitted, the new assignment would be written to the access control logic, which is the POLICY.JSON file. When the technician wants to access the UTILITY_DEVICES, the operational λ would check the access control logic file and grant the required permissions, if defined so.

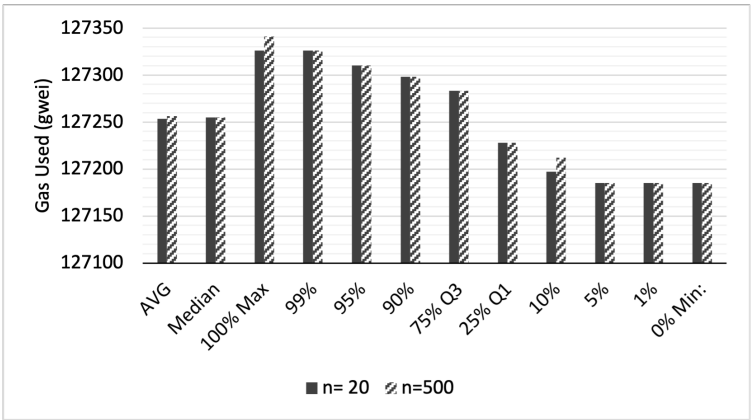

Figure 4.8: Statistical Summary of Gas Used

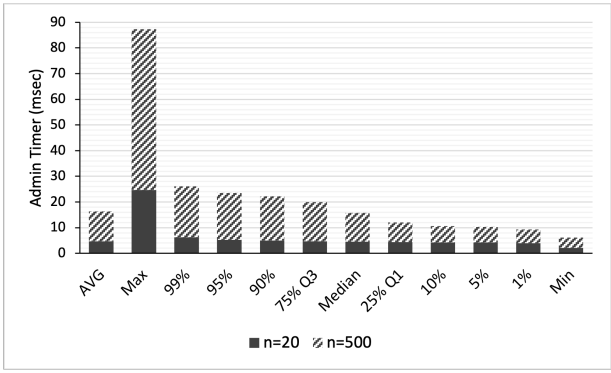

Figure 4.9: Admin Timer

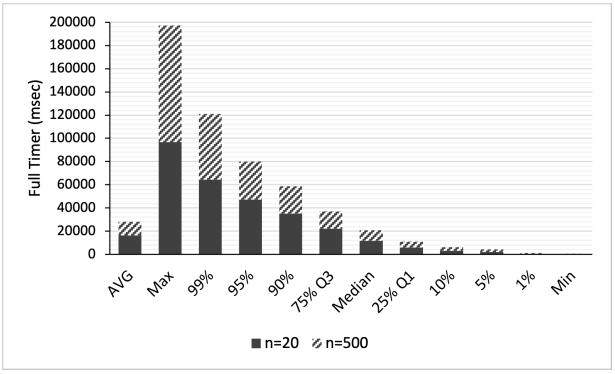

Figure 4.10: Full Timer

4.2.6 PEI: Implementation

Most of the proposed blockchain-based access control frameworks, which we briefly discussed in Section 4.2.2, have left their proposals at the conceptual level [89, 155] or evaluate them on either a locally built blockchain [135, 207, 217] or a testnet which is not using the same protocol as the Ethereum mainnet [87], so none of which provides a reliable and pragmatic assessment of practicality. In this paper, we validated our proposal of using blockchain for administrative access control enforcement by a proof-concept implementation. Further details are provided in the following sections.

Ethereum Blockchain

Ethereum could be viewed as a state machine in which a transaction would represent a valid transition between states [8]. A transaction is a single cryptographically signed instruction issued by an entity which is tied to an account. There are two types of Ethereum accounts, externally owned accounts (EOA) which belong to an external user and controlled by a private key, and contract accounts which contain and are controlled by the code. Transactions collected into blocks which are chained together via cryptographic hashes to create the blockchain. Each block should be distributed and agreed upon by every node in the network before being added to the chain, using a consensus algorithm. Current version of Ethereum uses proof-of-work, a.k.a PoW, as its consensus algorithm.

It is not a preferable choice to develop and test the smart contracts on the primary public blockchain of Ethereum, a.k.a mainnet, for two reasons. First, because of the immutable nature of blockchain, changing the smart contract code would be a challenging issue as rewriting at transaction level within the blocks is still in its infancy [59]. Second, Ethereum has its own cryptocurrency called *ether* and an internal currency called *gas* to pay the fee of transaction on Ethereum which is proportional to the amount of required computational effort. As any transaction with the mainnet needs gas to be run, buying the real ether (ETH) to provide the gas is prohibitive for testing purposes. Therefore, multiple test networks, a.k.a testnets, have been introduced to test smart contracts before deploying them on the mainnet.

We deployed a smart contract on the Ropsten testnet which is the official Ethereum testnet [11] that uses the same consensus protocol, PoW, as the mainnet. Because many users test their applications on this test network before deployment on the real chain, we recognize it to be a better simulation of a real-world scenario [126]. Therefore, we consider our results to be close enough to the real-world scenario of using the main Ethereum network. Other testnets, such as Rinkeby, Kovan and Görli are using proof-of-authority, a.k.a PoA, which is more time and energy efficient, but different from the mainnet consensus protocol. Therefore, we consider those testnets as nonviable and unreliable to represent the mainnet.

Smart Contract

We implemented administrative access control policy in a single smart contract on the Ropsten blockchain, in which administrative units are predefined and different administrative controls have been coded as functions which would be triggered via transactions. Although the smart contract code is not modifiable after being deployed, it is possible to add/remove data to its stack. So, the administrator can define new tasks to be included in each administrative unit or remove a task from the list of prohibited tasks.

We programmed our smart contract in Solidity and tested it on Remix IDE [15] which is the official browser-based IDE for Ethereum. Administrative units and administrative tasks defined as separate *mapping* data structures in Solidity. To interact with a smart contract, which has been deployed on the Ropsten testnet blockchain, we used a WEB3 API facilitating interaction with the blockchain. We used Infura [13] as the connection point for the web3 API, which hosts some nodes of Ropsten and relays all transactions to the blockchain.

Experiment Setup

We simulated a use case provided in Figure 4.6 using AWS IoT Greengrass v1 which runs on a dedicated virtual machine with one virtual CPU, 2 GB of RAM and 20 GB hard drive. The virtual machine's operating system is Ubuntu 20.4.2 LTS and it is connected to a 1 Gbps network. Our AWS lambda code on the Greengrass is written in Python 3.8 and is running in a long-lived isolated runtime environment with limited RAM of 256 MB. Lambda function receives the administrative requests and connects to Infura API to check the transaction status and results, after they have been run on Ropsten testnet. The results would later be reflected on the user's phone via updating its shadow on the Greengrass by lambda function. These results would also be written into the POLICY.JSON file if the administrative request was submitted and approved by smart contract to update the access rules. The POLICY.JSON file would be referred to govern operational accesses in the smart home environment and is shared between operational and administrative Lambda functions and protected by a lock for concurrency control.

Implementation Results

Table 4.5: Statistical Analysis Results

	Gas Used (gwei)		Admin Timer (ms)		Full Timer (ms)	
	n=20	n=500	n=20	n=500	n=20	n=500
AVG	127254	127256	4.67	11.61	16234.85	12005.34
100% (Max Quantile)	127326	127341	24.75	62.51	96743.43	100602.62
99%	127326	127326	6.29	19.90	64345.05	56523.07
95%	127310	127310	5.22	18.34	47215.98	32847.43
90%	127298	127298	4.95	17.26	34990.34	23773.95
75% (Q3)	127283	127283	4.75	15.19	22015.52	14891.59
Median	127255	127255	4.60	11.20	11667.90	9035.50
25% (Q1)	127228	127228	4.38	7.62	6019.41	4845.63
10%	127197	127212	4.30	6.32	3171.00	2935.90
5%	127185	127185	4.25	6.07	2076.12	2294.55
1%	127185	127185	3.96	5.36	63.50	1097.91
0% (Min Quantile)	127185	127185	2.16	4.04	55.45	68.07

To evaluate the performance and practicality of our blockchain-based approach for administration of access, we implemented a proof-of-concept under the settings discussed in the previous section. That means each transaction has been sent to Infura and the raw transaction hash is being sent back to the user's phone and the Lambda function. Then, lambda waits for the transaction to be mined and afterwards updates the policy file based on the successful events in the transaction log. The results would also be sent to the user's phone. Experiments are done for a normal distribution with a 99.9% confidence interval. To synchronize timing of the local computer and time servers in case the administrator user wants to make policy changes when away from home, we used Chrony [5].

To avoid duplicates, each time a new rule has been administratively requested to be added to the POLICY.JSON file, we check the policy and add the new rule to the policy only if it has no replica in the current policy. We ran our experiments in two settings with the policy sizes of n=20 and n=500. In the first setting, we start with an original policy of size 20 and add one policy

in each experiment but keep the maximum size of the policy to be 21. After each experimental run, the original policy with 20 values is reinstated and any changes are dismissed. The second scenario starts with a policy size of 20 but grows incrementally with each policy submission. Both experiments were run for a total of 500 times, resulting in the first case a maximum of 21 policies, and the second case, a policy grew from 20 to the final size of 520.

All the statistical analysis results are provided in Table 4.5. A visual representation of two important metrics of time and cost are depicted in Figures 4.8, 4.9 and 4.10. Figure 4.8 shows the required gas for transaction mining on the Ropsten network. The used gas is the actual amount of gas which was used during execution. Gas prices are denoted in GWEI, which equals to 10^{-9} ETH. We calculated the monetary cost of each transaction to be 28 cents, based on the Ether price as of the time of writing this paper.

Based on the results depicted in Figure 4.9, the difference of average time required for adding a new policy rule (administrative action) would be highly affected by the policy size, which is an indicator of the lambda processing time. After a transaction has been successfully mined, Lambda checks the logs to search out the succeeded transactions. Then, it makes appropriate changes to the POLICY.JSON file and publishes the results to the USER/STATUS/UPDATE to inform the user about his/her administrative request. We call this time *Admin Timer* which is in order of milliseconds.

The Full timer in Figure 4.10 shows a complete cycle of an administrator submitting a request, to that request being mined, and the lambda function processing the results and updating as necessary. The average total time for an administrative task using blockchain could be estimated as 12.012 seconds. Although this time is unsatisfactory for end users in an operational model, it is quite acceptable for an administrative model, especially compared to the other administrative methods which may take in order of minutes, hours or even days to take effect. On the other hand, with Ethereum moving to Proof of Stake (PoS) as its consensus mechanism, the costs and timing are expected to decrease dramatically [7].

4.3 Discussion: Model/Architecture Properties and Limitations

In this chapter, we proposed an RBAC administrative model based on EGRBAC operational model in smart home IoT environments. We introduced the concept of administrative unit, which consists of a unique administrative role and a set of administrative tasks. Each administrative task corresponds to one of the ASSIGNMENTS in the operational model. The model has been extended to enclose another assignment relations of the model by introducing administrative sub-units and administrative sub-tasks. Therefor, the extended model would have appropriate parts for administrative tasks required to manage access changes due to adding a new device to smart home or imposed by changes in assignments in which device role is involved. We assume the set of regular user roles, administrative roles and device roles have been centrally managed in some way.

Proposed model's properties and restrictions recognized to be as follows:

Decoupled Assignment and Revocation Proposed authorization functions in our models decouple assignment and revocation. Therefore, any administrator user can conduct authorized grant/revoke assignments, provided that the function's preconditions are satisfied. This means there is no need that granting and revocation of a permission to be done by the same administrator.

Symmetric Assignment and Revocation Even though grant and revoke are decoupled as stated above, our authorization functions enable an administrator user to revoke a permission, which has been conferred previously by him/her, from a subject. Similarly the same administrator who revoked a permission is able to re-grant it in the future, as long as the administrator user holds the same administrative role.

Generalizability Although our model manages two of assignments in underlying operational model, it could be easily generalized to govern other assignments by defining extra administrative units, each of which would cover a new scope of administration defined as an administrative subtask.

Then, we presented an architectural enforcement of access administration for above-mentioned administrative model. The administrative model that we chose as our underlying policy model as per PEI framework [169], has its above-discussed characteristics and limitations which have been carried to our proposed architecture and implementation. Our architecture is based on Ethereum blockchain and hence is decentralized, auditable, and reliable. We backed up our proposed architecture with a proof-of-concept implementation. Our implementation results are reassuring that although the use of blockchain for operational access control is not promising, an administrative model could successfully utilize the benefits of blockchain. Additional characteristics of proposed framework as well as some security considerations have been briefly discussed as follows.

Transparency and Auditability Using Ethereum as a public blockchain provides full transparency of transactions as well as access to immutable history logs. Without blockchain, the context of access control decisions would no longer be available; however, blockchain logs provide the posterior auditability. Moreover, the smart contract remains publicly visible on the blockchain even if it would be disabled in the future; in such a case the actual contract remains on the chain but would be marked as not callable.

Privacy The distributed nature of the blockchain eliminates the concern of privacy leakage from a single point of administration. Using blockchain preserves the privacy of the smart home as a whole, because the smart contract is only accessible with users who have their private keys stored on their own devices. Privacy of each user in the smart home withholds through decentralized administration in the form of administrative units (AU); so that each user's privacy zone could be contained in a separate unit while that user has been the only user assigned to corresponding administrative role.

Smart Contract Security Benefits of creating decentralized applications (dApp) using smart contracts do not come without costs. As an account-centered model of transactions which is used to identify and communicate with smart contracts on Ethereum, authentication and authorization

failures may impose security risks to the system. Ethereum itself is vulnerability-prone, besides the security vulnerabilities which are introduced by unreliability of Solidity [51]. Different verification tools are proposed to analyze the security of deployed smart contracts, a survey of which could be found in [60]. We used Remix IDE [15] for our Solidity smart contract development which performs a static analysis during compilation and reports security vulnerabilities such as implicit typing/visibility, unchecked return values, deprecated constructs, and address checksum and where they occurred in the code. So, we could be sure that our smart contract code is free of vulnerabilities which are checked by Remix. Checking the developed smart contract with other available tools for security vulnerabilities could be considered as a future step, specifically if the contract is going to be generalized for larger environments.

Device-Cloud Communication As the proposed architecture presents cloud-enabled IoT devices, the security of AWS Greengrass and its communication with IoT devices has a great impact on the overall security of our system. IoT devices use X.509 public key infrastructure (PKI) certificates for authentication of devices to Greengrass which are securely tied to AWS IoT policies [6]. We considered best security practices recommended by AWS IoT according which we implemented our architecture in a way that each IoT device has a unique immutable identity stored on it, which would be used to agree on PKI certificates; so, there would be no hard coded credentials in lambda functions [16].

Proposed framework for smart home IoT environment still needs to be improved to address following restrictions:

Continuous Access Control and Mutability Considering the dynamics of a smart home IoT environment, as a multi-user multi-device environment we need to monitoring the access even after being granted, and sometimes need the immediate change [150]. Moreover, it is required to use access quotas as a consumable non-refundable amount of access to some resource. For instance, the available time for kids to access the PlayStation on a weekend needs to be monitored and access should be revoked immediately (continuous control) as it has been exhausted (mutability).

Conflicts of Access/Conflict of Interest We may not have policy conflicts in that our proposed

framework does not include any negative policies, instead to avoid a role from being conferred with specific permissions, we utilize prohibitive assignments. However, it is possible to have administrator interests conflicting. For instance, different homeowners adjust the smart thermostat to different temperature ranges. As users expected the conflicts to be resolved automatically based on the survey of access control needs in a smart home [214], it is highly recommended that a policy resolution algorithm to be incorporated in the access control framework of a smart home [187].

Continuous Usage Control Considering usage as practicing granted access rights by subjects on objects, it is required for dynamic environments to have continuous control over it. In many cases, enforcement of the access control necessitates an immediate change in permissions, e.g., when administrator revokes the access from a user who is currently utilizing it. There are administrative models proposed to enforce administrative decisions in a session-aware manner to satisfy this requirement [206].

Quota-based Access Enforcement Some access control requirements in a smart home environment may call for access quota. Access quota is a consumable amount of resource usage which is non-refundable. For instance, we may want to limit kids access to entertainment devices to one hour per day. Such requirements are irrefutable evidences that operational and administrative access control models should be able to handle quotas. Authors in [184] proposed a quota-based approach to address the consistency problem in ABAC environments.

Chapter 5: DEVICE-TO-DEVICE ACCESS CONTROL FOR IOT COLLABORATION IN SMART HOME ENVIRONMENTS

The Internet of Things (IoT) has been widely integrated in people's everyday lives. However, as an infrastructure of connected heterogeneous devices, IoT has not yet achieved the seamless integration of device-to-device collaboration. In a smart home IoT, smart devices which are also known as smart objects/things, include sensors or actuators through which they sense or change their physical surroundings. These IoT devices expected to exchange their collected data or status in certain circumstances, in spite of their heterogeneity, viz. working with different communication protocols, IoT platforms, middleware, data and semantics. Smart IoT environment in which device-to-device (hereafter D2D) communication happens seamlessly and directly among heterogeneous devices, is currently more of a vision than reality. Furthermore, in a smart home IoT scenarios of D2D interactions are inevitable for real-life home automation. Deploying appropriate access control models and mechanisms is of utmost importance as any unauthorized access to data could have a cascading violation of privacy, safety and security of users. In this chapter, we propose a novel device-to-device access control model in the smart home IoT. Our approach relies on message passing as the paradigm for device-to-device interactions. We further introduce actions and scenarios reflecting the chain of events in the smart home context, which facilitates scenario-driven attribute-based access control. Each Scenario is triggered by triggering events, based on previously set administrative definitions. We define a totally ordered sets of triggering events using priorities to enable conflict resolution for devices which may run into conflicting commands delivered though messages in different ongoing scenarios . The viability of the proposed approach is substantiated via a formal model and an enforcement architecture, backed up by a proof-of-concept implementation which affirms a trade-off between required authorization and efficacy. Potential future challenges are explored in the context of smart home IoT platforms. Results of this research has been submitted to the following conference:

1. Mehrnoosh Shakarami, James Benson, and Ravi Sandhu, "Scenario-Driven Access Control

for Device-to-Device Collaboration in Smart Home IoT", 36th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec'22), July 2022.

5.1 Motivation

Considering home IoT devices as an ecosystem with inter-communications provides a holistic perspective toward home automation and brings added convenience. However, it introduces potential risks to the safety and privacy of home users. For instance, an attacker can compromise a device to misuse it as a breaking point for unauthorized access to the network to which it is attached. For instance, a compromised device by an attacker can maliciously act as a breaking point for unauthorized access to the network to which it is attached. Thereafter, the hacker can break into the home and unlock the doors while it is vacant, or change the air-conditioning mode to heat in summer hot days [130, 218]. Although there are handful of studies focused on individual parts of the smart home IoT, e.g., device authentication [122, 136, 139], communication protocols [79, 84, 167], and home automation applications [47,74,118], research body on security of interactions in smart home is quite scarce. In this context, securing D2D interaction through authorization of the flow of information among devices is of utmost importance. A maliciously overtaken device by an attacker may send control information to manipulate other devices' operations, for instance requesting sensitive recordings of indoor cameras.

Access control is a crucial requirement in a smart home IoT to regulate the access and authorize communications among IoT devices. Although there is a research body on regulating user-to-device (hereafter U2D) access in smart homes, no previous work has been devised to regulate the device-to-device communications through specification of an access control model. Our main contribution is to formulate an access control model which governs authorized flow of information among home IoT devices using Attribute-Based Access Control (ABAC). This is the first research to do so in the smart home context, to the best of our knowledge. An ABAC policy enables capturing the dynamics and fluidity of the smart home environment by entailing contextual/environmental attributes such as time and location. Moreover, ABAC facilitates defining authorization rules based

on attributes of devices which are engaged in D2D communication. Also, using ABAC for defining D2D access control policy brings its inherent flexibility of defining authorization rules based on attributes of devices which are engaged in D2D communication. This enables us to define prerequisite conditions in the form of specific attribute values of communicating devices to regulate the flow of information among them.

One contribution presented in this chapter is utilizing the message passing paradigm for D2D access control, which is especially suitable for this domain. We inspect the contents of exchanged messages in D2D communications as attributes of the messages, based on which the authorization rules in the proposed access control model would be defined. In its essence, our approach defines a set of authorized flow of messages between devices through establishing access control rules based on sender/receiver device attributes as well as context attributes. Prohibited D2D communications (negative permissions) could implicitly be inferred by exclusion of the desired policy rule that would otherwise authorize that communication if included.

Another contribution is extending our model to include scenarios as a sequence of actions that may occur in the home IoT environment. Different sequence of actions at the smart home IoT in our model would be initiated by a trigger, i.e., an event or a set of events in the smart home. As any D2D action in the smart home IoT is considered to be done via message communication between devices, we conclude each trigger would initiates a set of message passing among devices in the smart home IoT environment. Then actions would be coupled with their triggering events to define scenarios. By defining priorities among triggers, we equipped our model with conflict resolution. So, if a device is involved in conflicting ongoing scenarios, the conflict could be resolved without human intervention.

Configuring the of legitimate D2D communications, and corresponding access rules is specified by human administrators, namely homeowners or parents. For example, a homeowner may prohibit any communication of indoor cameras' recording data with other home devices or the homeowner may prefer any suspicious motion in their vacant house to be reported. We consider such tasks to be administrative ones based on which desired flow of information would be defined,

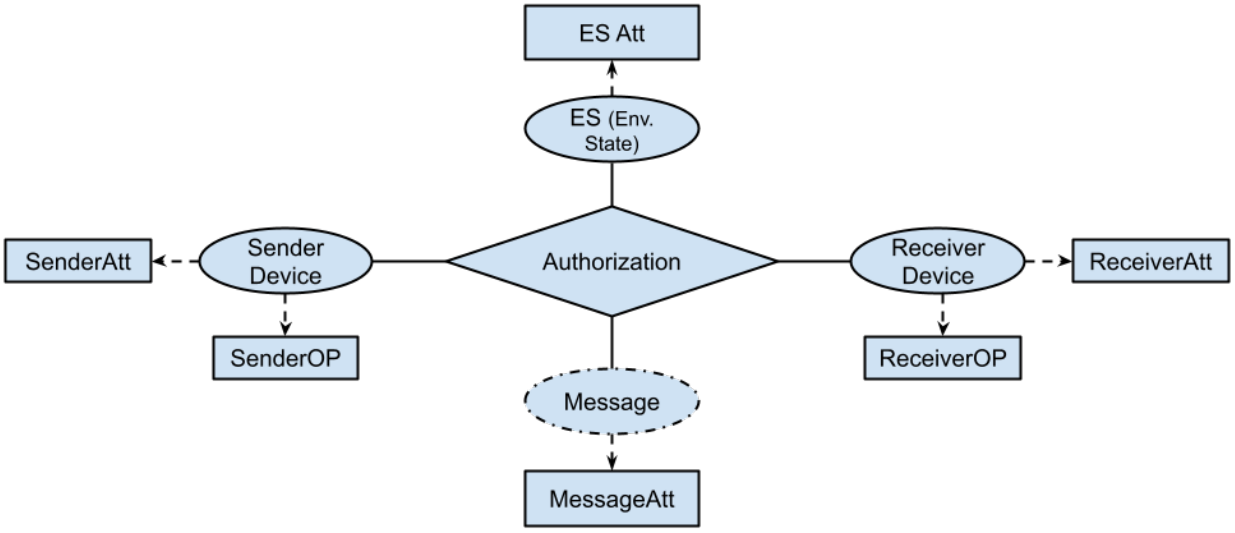

Figure 5.1: Device-to-Device ABAC Model

that precede the regulation of D2D access control at operational level. Thereafter, no user intervention is required for our mediation of proposed D2D access. We acknowledge the complications concerning the evolving D2D communication, e.g., different platforms, communication protocols and data models, and accordingly design a consolidated access control approach in this context. We consider direct communication of heterogeneous IoT devices out of scope [200], and aim our work to serve as a gateway-enabled initiative in the scarcely investigated area of D2D access control.

5.2 Message-Based D2D ABAC Authorization Model

In order to design an appropriate access control framework for any IoT environment, capturing the context which adapts the dynamicity of the IoT ecosystem is quite required [160]. The risk of insecurity due to unauthorized access varies by the context in which the IoT device operates. An example of risk could be an IoT device with weak security providing unauthorized access to the network to which it is attached or serving as a bot in attacker's control in different attack scenarios, such as Denial of Service [130]. The context of an entity could be interpreted as any information to characterize its situation [18]. For an IoT device, the context could be its location, battery status and owner. Contextual access control is mostly required for access control in a smart home IoT environment [94], There are some access control proposals in IoT environments which

design or create the IoT authorization policies while capturing the context relying on semantic web technologies [68], RBAC extensions [216], ABAC [53], OrBAC [86] or combination of different access control paradigms [63].

There are proposals in IoT environments which utilize messages for control and communication [71], authentication [136], discovery and configuration of new IoT devices [153] and routing [33]. There also has been some research work on securing communication using access control models [56]. Alshehri and Sandhu [23] identified the need for controlling data and communication in IoT environments. A general conceptual model for attribute-based communication control has been developed by Smriti et al. [40]. Our proposed model is the first access control model based on the message passing paradigm in IoT environments for authorization of device-to-device communications.

In this section, we design to capture the IoT device context via the contents of communicated messages in device-to-device interactions. In our approach, a message is a distinct structured entity being communicated between two IoT devices. Message attributes are derived from its content. The contents of a message are structured in a *(key,value)* format and reflect the communication context, including the purpose of communication which is reflected in the message classification followed by its corresponding attributes in the message. Unique to our approach is to propose a message passing paradigm for IoT device-to-device access control. No two IoT devices would be able to communicate to each other, unless there is a corresponding authorization rule defining their possible type of communication. In other words, the set of authorized device-to-device communications have been defined through establishing access control rules based on attributes of environment, sender and receiver devices, and the transmitted message. Notably, for the first time our model presents the IoT D2D communication to be without human intervention.

5.2.1 Conceptual Model

Our conceptual ABAC model is depicted in Figure 5.1. There are two IoT device endpoints in each communication, subject to authorization, determined as *sender device* and *receiver device*.

Each of two communicating endpoints (either sender or receiver) has its own assigned attributes, such as ID, type, location, etc. The attributes of sender/receiver devices are represented as *SenderAtt*/*ReceiverAtt* respectively. Moreover, for each endpoint device there is a set of available functionalities, defined by its manufacturer, and represented as *SenderOP*/*ReceiverOP* as depicted in Figure 5.1. Message introduced in our proposed model as a new element of D2d access control. Message attributes are extracted from its content and present the context of communication, i.e., the attributes of the environment (smart home) in which messages are transmitted. Message attributes are represented as *MessageAtt* in Fig 5.1. Examples of message attributes could be its type, location, requested attribute/action from the receiver device, etc. As messages are transitory elements of the system, they do not exist until being sent/received by corresponding devices. Thus, we depicted the message entity in a dotted circle in Figure 5.1.

To be in accord with the highly dynamic nature of IoT smart home, we consider Environment state as an entity of our system represented as *ES* and utilize its associated attributes, a.k.a. *ESAtt* for access control decisions. Examples of ESAtt include daylight, time of the day, weather conditions, etc. *Authorization* function is defined based on entities' attributes as depicted in Figure 5.1. When one IoT device requests to communicate with another IoT device, the authorization function, which has been defined by *CheckAccess* predicate in our formal representation, allows or denies the request.

5.2.2 Formal Model

Proposed model's formalization has been presented in Table 5.1. The basic components of the proposed model are discussed below.

Core Components. Include the set of IoT devices (D), their available operations (OP), the set of environment attributes presenting the environment state (ES). D, ES and M (messages) are considered as entities (E) in our model. Device operation assignment (DOA) is a vendor-defined relation which determines the set of available functionalities to a device.

Attribute Functions. We consider each entity's attribute value type, represented as $AttValueType$

Table 5.1: Message-Based ABAC Model Formalization

Core Components

$-D$ is a set of smart home IoT devices deployed by homeowner.

$-OP$ is a set of operations available on different devices in the system (manufacturer specified).

$-ES = \{current\}$ is the singleton set, representing the environment state at the current time instant.

$-Ent = D \cup ES \cup M$ is the set of entities in the system, where the set of messages M is defined below.

$-DOA : D \rightarrow 2^{OP}$ is a one to many relation which associates a device to its available operations as specified by the device manufacturer.

Attribute Functions

$-DAA, EAA$ are respectively sets of attribute functions which associate a device or the current environment state with attribute values.

$-attValueType : DAA \cup EAA \rightarrow \{atomic, set\}$

$\forall att \in DAA \cup EAA,\ Range(att)$, is the attribute range, a finite set of atomic values.

$-$Each $att \in DAA \cup EAA$ maps a device/environment to a single *atomic* value or to a finite *set* of values, as follows:

$$-att : DAA \cup EAA \rightarrow \begin{cases} Range(att) & \text{if } attValueType(att) = atomic \\ 2^{Range(att)} & \text{if } attValueType(att) = set \end{cases}$$

$$-attAssignType : DAA \cup EAA \rightarrow \begin{cases} static & \text{set/changed via administrative actions} \\ dynamic & \text{set/changed automatically by deployed sensors in home IoT} \end{cases}$$

Message and Message Functions

$-M = \{m\}$ is the set of all messages in the system.

$-m = \{(att_1, value_1), (att_2, value_2), ..., (att_n, value_n)\}$, represents any single message in the system with n different attributes, each of which is indicated as a $(key, value)$ pair.

$-typeSet = \{"query", "command", "info"\}$ is a mandatory first attribute in every message which indicates its *type* and thereby the rest of message attributes.

$-$For each $m \in M$, we assume the first attribute determines the type of the message: $att_1 \in typeSet$

$-typeSetAtt : M \rightarrow 2^{DAA} \cup 2^{DOA}$, is a function which indicates the set of attribute keys required to be communicated based on the message type, supposed to be communicated via $\{att_2, ..., att_n\}$ in each message.

Check Access Predicate

$-CheckAccess$ is evaluated when a sender device (s) tries to send a message (m) to a receiver device (r) in context of current environment state ($current$) and is evaluated based on following formula:

$-CheckAccess(s : D, m : M, r : D, current : ES) \equiv$

$CheckAtt(s : D, m : M, r : D, current : ES) \wedge Authorization(s : D, m : M, r : D, current : ES)$

$$-CheckAtt = True \iff typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = "query" \\ \in DOA(r) & \text{if } m.value_1 = "command" \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = "info" \end{cases}$$

$-Authorization(s : D, m : M, r : D, current : ES)$ is a logical proposition which could be evaluated to either True or False and is created using following policy rules.

$-p \equiv (p) \mid \neg p \mid p \wedge p \mid p \vee p \mid \exists x \in set.p \mid \forall x \in set.p \mid set\ \Delta\ set \mid atomic \in set$

$-\Delta \equiv \subset | \subseteq | \not\subseteq | \cap | \cup$

to be either *atomic*, which means it would have one of the values of its range at each moment, or a set, which could be assigned to a subset from its range. Moreover, any device attribute is either *static*, i.e. its value is fixed and statically defined by its owner/manufacturer, or *dynamic*, i.e., its value could change as the side effect of message communication with other devices or the operations done by the device itself. The assignment type for each attribute is represented as $AttAssignType$ relation. EAA and DAA denote the environment/device attribute assignment relations which associate attributes to the environment state and device entities respectively.

Message and Message Functions. M is a set of messages. Each message is a set of *n* attributes in the form of *(key,value)* pairs, communicated between two devices. Multi-messaging and message broadcasts are out of scope. As messages are transient entities, its sender's ID and receiver's ID are not contained in the template of the message definition. The message type is defined as a mandatory first attribute of every message, which determines the rest of required attributes to be included in the message. Every message belongs to one of the classifications indicated in *typeSet*, restricted to *"query"*, *"command"*, and *"info"*. Message type is reflected in the first pair of its attribute represented as $(\text{"}type\text{"}, value_1)$ in which $value_1$ could be one of the predefined types in *typeSet*.

A message of type *"query"*, inquires into the value of attributes of the receiver, for instance the outdoor camera may send a query message to the door lock to know its "locked" attribute value. This message might be frequently communicated, to monitor any changes in the attribute's value. The frequency of communication depends on the device, its state changes, and the commands issued. A message with type *"command"* orders the receiver device to perform an operation towards its environment, for example a security camera may command the door lock to be locked. An *"info"* message informs the receiver about the values of some of the attributes of the sender. Any message of *"query"* or *"command"* requires receiving back a message of type *"info"* in response, which provides the requested attribute value or acknowledges the command's fulfillment. An *"info"* message needs no response.

Check Access Predicate. This function would be evaluated for each message communication to

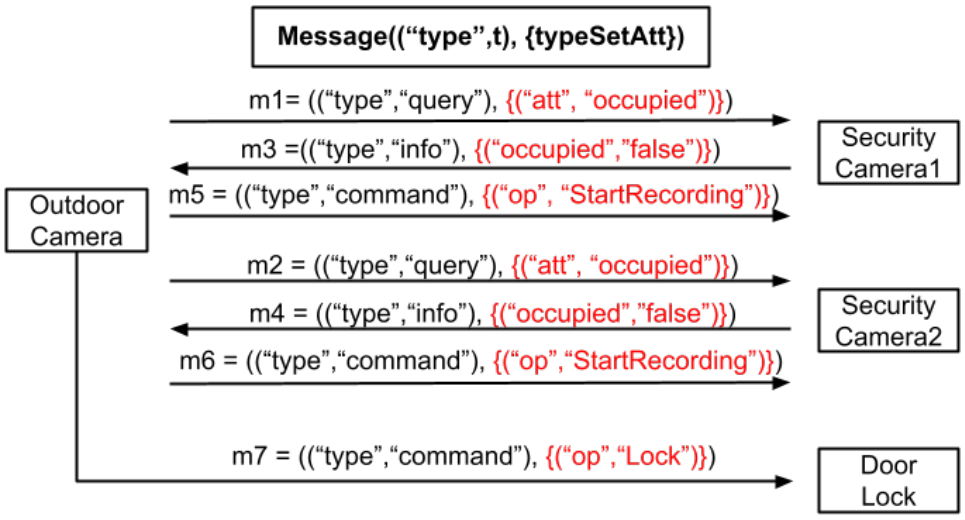

Figure 5.2: Smart Home Use Case for Device-to-Device Communication

TRUE or FALSE which respectively indicates allowance or denial of communication of specified message from sender to receiver. In order to authorize a message communication between a sender device and a receiver device, two functions would be checked by *CheckAccess* function. *CheckAtt* checks the *feasibility* of the message which is intended to be communicated, that is determined based on message type. If a message is of type *"query"*, the requested attributes by sender must be a subset of attributes defined for the receiver device through *DAA* relation. Similarly, for the *"info"* messages, the communicated set of attributes has be a subset of sender device's attributes, defined via *DAA*. For messages of type *"command"*, the operation which senders ask the receiver to do must be a functionality which is available to the receiver and defined through *DOA* by its vendor/manufacturer. For instance, if a sender device asks a receiver's device to do an operation which is not defined for the receiver, this function returns "false".

5.2.3 Smart Home Use Case

In this section we present a use case to show how the components of our model should be configured to enforce D2D access control in a smart home IoT environment without any need for human intervention. Consider a smart home IoT with a single owner (say Alice), which comprises a smart door lock, two indoor smart security cameras and a smart outdoor camera. We assume all

indoor and outdoor cameras are equipped with presence sensors, so the outdoor camera is able to detect Alice arrival/departure and indoor security cameras could detect whether anybody is home. We represented a simple scenario in which home automation through the proposed access control model is authorized without any need for human intervention.

Table 5.2: Message-Based D2D Access Control: Smart Home Use Case

Begin of Table

Core Components

$D = \{OutdoorCamera, SecurityCamera_1, SecurityCamera_2, DoorLock\}$
$OP = \{Lock, Unlock, StartRecording, StopRecording, TurnOn, TurnOff\}$
$ES = \{current\}$
$DOA = \{(SecurityCamera_1, \{StartRecording, StopRecording\}),$
$(SecurityCamera_2, \{StartRecording, StopRecording\}),$
$(DoorLock, \{Lock, Unlock\}), (OutdoorCamera, \{StartRecording, StopRecording\})\}$
$DAA = \{(SecurityCamera_1, \{id, type, location, recording, occupied\}),$
$(SecurityCamera_2, \{id, type, location, recording, occupied\}),$
$(OutdoorCamera, \{id, type, location, recording, incident\}), (DoorLock, \{id, type, location, locked\})\}$
$EAA = \{day, time\}$

Attribute Functions

$id(SecurityCamera_1) = "sc1"$, $type(SecurityCamera_1) = "camera"$,
$location(SecurityCamera_1) = "indoor"$, $recording(SecurityCamera_1) = \{"true", "false"\}$,
$occupied(SecurityCamera_1) = \{"true", "false"\}$
$id(SecurityCamera_2) = "sc2"$, $type(SecurityCamera_1) = "camera"$,
$location(SecurityCamera_1) = "indoor"$, $recording(SecurityCamera_2) = \{"true", "false"\}$,
$occupied(SecurityCamera_2) = \{"true", "false"\}$
$id(OutdoorCamera) = "oc1"$, $type(SecurityCamera_1) = "camera"$,
$location(SecurityCamera_1) = "outdoor"$, $recording(OutdoorCamera_1) = \{"true", "false"\}$
$incident(OutdoorCamera_1) = \{"coming", "leaving"\}$
$id(DoorLock) = "dl1"$, $locked(DoorLock) = \{"true", "false"\}$, $type(DoorLock) = "lock"$,
$location(SecurityCamera_1) = "mainEntrance"$,

$$\begin{cases} attAssignType = static, & \text{if } att \in \{"id", "type", "location"\} \\ attAssignType = dynamic, & \text{otherwise} \end{cases}$$

Message and Message Functions

$M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$
$m_1 = \{("type", "push"), \{("att", "occupied")\}\}$
$m_2 = \{("type", "push"), \{("att", "occupied")\}\}$
$m_3 = \{("type", "push"), \{("occupied", "false")\}\}$
$m_4 = \{("type", "push"), \{("occupied", "false")\}\}$
$m_5 = \{("type", "com"), \{("op", "StartRecording")\}\}$
$m_6 = \{("type", "com"), \{("op", "StartRecording")\}\}$

Continuation of Table 5.2

$m_7 = \{(\text{"}type\text{"}, \text{"}com\text{"}), \{(\text{"}op\text{"}, \text{"}Lock\text{"})\}\}$

Check Access Predicate

$-CheckAccess(s : D, m : M, r : D, current : ES) \equiv$
$CheckAtt(s : D, m : M, r : D, current : ES) \wedge Authorization(s : D, m : M, r : D, current : ES)$

$$-CheckAtt = True \iff typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = \text{"}query\text{"} \\ \in DOA(r) & \text{if } m.value_1 = \text{"}command\text{"} \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = \text{"}info\text{"} \end{cases}$$

$-Authorization(s : D, m : M, r : D, current : ES) \equiv q_1 \vee q_2 \vee q_3 \vee q_4$

$$q_1 = \Big[\Big(m.att_1 = \text{"}query\text{"}\Big) \wedge \Big(typeSetAtt(m) \in \{\text{"}recording\text{"}, \text{"}occupied\text{"}\}\Big) \wedge$$
$$\Big(type(s) = type(r) = \text{"}cameras\text{"}\Big) \wedge \Big(location(s) = \text{"}outdoor\text{"}\Big) \wedge \Big(location(r) = \text{"}indoor\text{"}\Big)\Big]$$

$$q_2 = \Big[\Big(m.att_1 = \text{"}info\text{"}\Big) \wedge \Big(typeSetAtt(m) \in \{(\text{"}recording\text{"}, \text{"}occupied\text{"}\}\Big) \wedge$$
$$\Big(type(s) = type(r) = \text{"}cameras\text{"}\Big) \wedge \Big(location(s) = \text{"}indoor\text{"}\Big) \wedge \Big(location(r) = \text{"}outdoor\text{"}\Big)\Big]$$

$$q_3 = \Big[\Big(m.att_1 = \text{"}command\text{"}\Big) \wedge \Big(typeSetAtt(m) \in \{\text{"}StartRecording\text{"}, \text{"}StopRecording\text{"}\}\Big) \wedge$$
$$\Big(type(s) = type(r) = \text{"}cameras\text{"}\Big) \wedge \Big(location(s) = \text{"}outdoor\text{"}\Big) \wedge \Big(location(r) = \text{"}indoor\text{"}\Big)\Big]$$

$$q_4 = \Big[\Big(m.att_1 = \text{"}command\text{"}\Big) \wedge \Big(typeSetAtt(m) \in \{\text{"}Lock\text{"}, \text{"}Unlock\text{"}\}\Big) \wedge$$
$$\Big(type(s) = \text{"}cameras\text{"}\Big) \wedge \Big(type(r) = \text{"}locks\text{"}\Big) \wedge \Big(location(s) = \text{"}outdoor\text{"}\Big) \wedge$$
$$\Big(location(r) = \text{"}mainEntrance\text{"}\Big)\Big]$$

End of Table

Figure 5.2 represents our use case in which the messages are shown without *sender* and *receiver* fields, as these fields are determined any time a device wants to send a message to another device. In this use case, sender and receiver are shown in the figure in the period of time during which our use case is happening. The outdoor camera takes the initiative when it detects Alice is leaving. It then checks the *"occupied"* attribute of indoor security cameras via sending query messages with *typeSetAtt* specified as $\{(\text{"}att\text{"}, \text{"}occupied\text{"})\}$ in m_1 and m_2. If the house indicated to be vacant

(*"occupied"=false*) by all security cameras, then outdoor camera sends *"command"* messages to all cameras to start recording, indicated as $\{(" op", " StartRecording")\}$ in the *typeSetAtt* of the messages m_5 and m_6. We assume our devices rely on the authorization done though CheckAccess predicate and would run the commanded operation. Moreover, outdoor camera sends a command message to the door lock to be locked, indicated as $\{(" op", " Lock")\}$ in the *typeSetAtt* of the messages m_7.

As soon as Alice is detected to be back by the outdoor camera, it detect *arrival* incident which consequently sends appropriate messages to involved devices, so the recording would be stopped, and the door unlocked. Our proposed access control model could be configured as shown in Table 5.2 to implement this use case. Notably, Check Access Predicate depicts the propositions to define authorized message communications among devices. If any device asks for any attribute of the other device, which has not been included in the check access predicates, that communication would be denied. For instance, if the outdoor camera requests the current vacancy status of the home by sending a *query* message requesting the *occupied* attribute, the request would be authorized. However, if it asks another attribute which is not permitted in the Check Access predicate, that message would not be authorized in Check Access, and hence would not go through. As another example, if the indoor security cameras try to send a command message to the door look and order it to Unlock, this communication would be denied as it is not included in the set of authorized communications in Authorization function under Check Access predicate. So, the message would not be sent though.

5.2.4 Threat Model

Smart home IoT devices are adopted by owners to enhance their lives security and convenience. Nevertheless, these devices' susceptibility to cyber attacks could make them disturbing security holes. For instance, a compromised security camera could be abused to send its recordings to the unknown outsider, or an attacker may lock you out of your home in case the door lock is hacked. A hacker with access to your thermostat could fiddle with it, causing your HVAC system

to malfunction. Worse yet, a hacker could crank up the oven and cause a house fire while you are away from home[1]. Some of the possible attack scenarios, a.k.a attack trees, in a smart home environment are discussed in [82, 132]. Everything considered, providing security in complex and dynamic environments such as smart homes remains a noteworthy challenge. We adopt the Dolev-Yao (DY) threat model [62] in which communicating endpoints cannot be considered as trusted nodes in the network. According to the DY model, an adversary can tamper with the data through modification, deletion, or insertion of fake information as the communications rely on wireless medium. As a confirming evidence of insecure communications, a 2020 IoT threat report announced 98% of all IoT traffic was not encrypted, thus vulnerable to security threats [2].

We are making the following assumptions for the D2D communication in the smart home in our research:

1. As many IoT devices are not IP-enabled, using a gateway (GW) node in the network is inevitable [200]. We assume the GW node in our model is trustworthy and available, which is a common assumption [156].

2. Attacker is considered to be an outsider to the network with the goal of obtaining illegitimate access to available functionalities/operations of smart home IoT devices.

3. We do not consider adversaries to have physical access to IoT devices.

Our access control approach provides a defense-in-depth prevention/protection against any outsider attack, which is aiming at unauthorized access to an IoT device's operation or information. Our model restricts the set of authorized messages being communicated among IoT devices as there is a subset of one IoT device attributes/operations which are accessible to another device. Suppose a well-positioned attacker in the network eavesdrops the communications and is able to impersonate as a legitimate IoT device at home using the known vulnerabilities of the device platform. Our model blocks the attacker's request for arbitrary information s/he may desire to acquire from other devices at home. Therefore, it would be arduous for an attacker with no knowledge of

[1]https://www.bobvila.com/slideshow/the-10-biggest-security-risks-in-today-s-smart-home-53081
[2]https://app.hushly.com/runtime/content/xVukSNKffbmoOef2

established policy rules, to realize which information is available to the cracked device to request from other home devices, yet that information may be of no use to the attacker. Our model is a barrier established, a.k.a a defense-in-depth strategy [3].

5.3 Scenario-Based D2D ABAC Authorization Model

Access decisions in a smart home may conflict due to conflicting consequences of different triggered actions, or simply a conflict of interest among owners of IoT devices, since IoT devices could be shared among smart home users. As an example, consider following two policies:

1. A rule is defined to command the door lock to be locked when an outdoor camera detects the owner of the house is leaving.

2. Another rule is defined to unlock the door when smoke is detected by fire detectors in the house.

Suppose the homeowner, say Alice, has left the house and the outdoor camera has sent a *command* message to lock the door. Alice is not back yet, so the outdoor camera would not send the *finish* message and the door lock remains locked. If a fire is detected at the house, then the fire detector sends a *command* message to unlock the door, which conflicts with the current lock command in effect. This conflict happens because the same device received two messages including conflicting commands. In this section, we propose a scenario-driven access control model to resolve a class of conflicts which happen because the same device received two messages including conflicting commands. Another use case is discussed in detail in the following sections.

5.3.1 Conceptual Model

Conceptual representation of our model is depicted in Figure 5.3. As stated previously, a conflict may happen due to conflicting *command*-typed messages received by the same device, initiated by events which we call *triggering events*. In fact, conflicting messages are the consequences of

[3] https://csrc.nist.gov/glossary/term/defense_in_depth

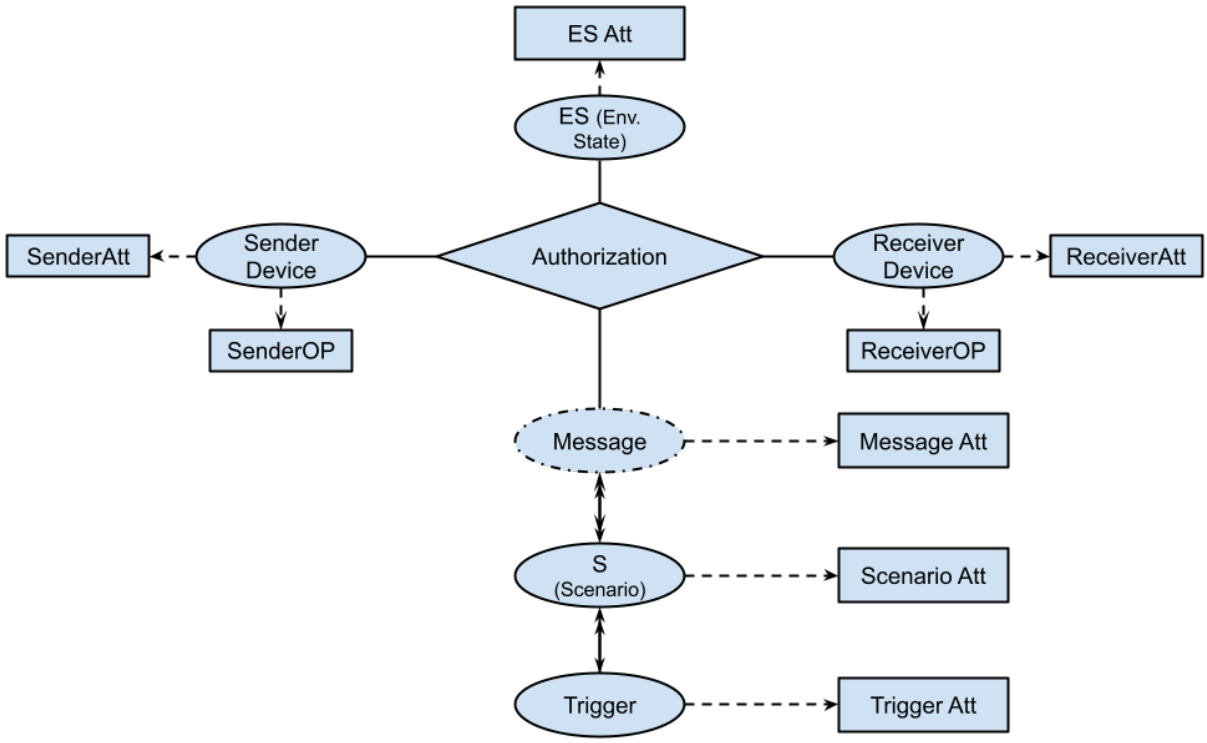

Figure 5.3: Device-to-Device Scenario-Driven ABAC Model

two conflicting events at home, viz. leaving the house by owner and then smoke detection by fire alarm, which we call *triggering events*, that is to initiate a set of messages to be communicated among IoT devices, so an appropriate action to be done in response. We use trigger and triggering event interchangeably in this dissertation.

Definition 5. A triggering event is a specific event or status from the IoT device's operation, e.g., the door is opened or the user is leaving home.

Each trigger would initiate a set of actions, which we collectively call an *action*. As we consider any action in smart home IoT to be done via message communication, we define an action as a set of messages communicated among different devices.

Definition 6. An action *a* indicates a message *m* being communicated from a specific sender *s* to a specific receiver *r* and defined as a triplet of $a = (s : D, m : M, r : D)$. *Action* is a set of actions predefined by the administrator/homeowner in the system.

A trigger may initiate a set of actions in the smart home, which we collectively call a *scenario*. So, each scenario is considered as a set of actions done in the smart home environment. A Trigger could provoke one or more scenario(s) in the smart home, i.e. the set of actions which has to be consequently executed in the home. We define priorities as a binary relation among triggers which conceptually reflect the importance of a trigger and its consequences.

Definition 7. priority is a totally ordered set relation, depicted as $(pr, \prec)$ between any two triggering events tr_i and tr_j and is reflected in their (administratively) assigned priority values. So, for any two triggers tr_i and tr_j, it is either $(tr_i \prec tr_j)$ or $(tr_j \prec tr_j)$.

The priority of each scenario would be the same as its triggering event's priority. Consequently all messages contained in the actions of that scenario would have the same priority. so any discord among their constituent messages (accordingly actions) could be resolved by a simple rule of priority: A message with higher priority would go through, and other conflicting messages would be disregarded. Thereafter, the precedence of actions would be determined based on their containing scenario's priority. A scenario is called *active* when it has been initiated via the happening of its corresponding triggering event.

Definition 8. A scenario is defined as a set of actions to be done in the smart home. Besides, the triggering event which provoked the scenario, its priority, and its *active* status would also be part of each scenario's definition, as shown in Table 5.3.

Our conceptual model is represented in Figure 5.3. Along with other elements of our message-based model, we add another element *Trigger*, as defined in Definition 5. Each trigger has attributes shown as *TriggerAtt*, including its administratively assigned priority, the set of device attributes/operations defining that trigger, etc. *Scenario* is another new element, which includes a sequence of actions in the form of message communications between IoT devices. Each scenario has its own attributes, shown as *ScenarioAtt*, including the set of scenario's contained actions, its priority, its trigger,its active status, and its assigned id. There is a one-to-many relation between *Scenario* and *Trigger*, which indicates one trigger may initiate multiple scenarios. This relation is defined through TriggeringEvent-Scenario Assignment relation ($TeSA$), as shown in Table 5.3.

As scenarios are defined at administrative level, we consider providing home admins to define the consequences of one trigger as multiple scenarios (one-to-many relation). This equips our model with higher flexibility and facilitates privacy considerations. Any authorized communication has to be included in the set of authorized actions by home administrators. Also the set of triggering events and thereby provoked scenarios has to be defined. Each trigger has to be assigned with a priority, which would be assigned to its consecutive scenarios, their contained actions and thereby messages. When a device wants to communicate with the other device, in order to craft the message's priority a function, namely $msgPrA$, would be called. If the message's *typeSetAtt* matches with a message in an action, which itself is included in an active scenario, then the priority of the message would be the same as the scenario's priority. If the message matches with more than one scenario, the highest priority would be assigned to it. On the other hand, in case of no match, the lowest defined priority in the system would be assigned to the message.

Anytime an IoT device sends a message to another IoT device, the Check Access predicate would be evaluated. Check Access includes three functions. CheckAtt which detects the feasibility of communication. CheckPriority examines the message priority and decides if the message should go through or be disregarded. Authorization defines the set of authorized flow of communications based on attributes of environment, sender and receiver device, and message attributes. Any conflict would be resolved by authorizing the message communication with higher priority.

5.3.2 Formal Model

Scenario-based model formalization has been represented in Table 5.3. Extra Components, compared to the message-based model, are as follows.

Core Components. An extra core component in the scenario-based model is triggering event set (TE), which could be any change in the IoT device(s)' attributes or an action which has been done by an IoT device. The set of entities include the set of devices, environment state, messages, triggering events and scenarios.

Attribute Functions. $CurrentOP$ is a new function indicating the operation each device is doing

at the current instant of the time. $CurrentPR$ is the priority of the current operation which is being run by the device at the current instant of time, basically inferred to be the same as the priority of the command message which made the device run that operation. For each device, $conflict$ defines a set of conflicting pairs of functionalities available to that device. For instance, $(Lock, Unlock)$ should be included in the $conflict(doorLock)$, as those functionalities cannot happen simultaneously, so they are conflicting. Priority is defined as a totally ordered set which defines a strict order between any two triggers in the system. Each trigger would be administratively assigned with its priority, which then at operational level is retrieved via prA function.

Table 5.3: Scenario-Based D2D Access Control Model

Begin of Table
Core Components $-D$, OP, ES have the same definition as Message-Based Model. $-DOA$, DAA, EAA remain the same as Message-Based Model. $-TE \subset D \times \{2^{DOA} \cup 2^{att(d:D)}\}$, is a set of triggering events. att has the same definition as in Table 5.1. $-Ent = D \cup ES \cup M \cup S \cup TE$ is the set of entities in the system, where the set of messages M and scenarios S are defined below. **Attribute Functions** $-currentOP : D \times \{current\} \rightarrow 2^{DOA} \cup \emptyset$, is the operation each device is doing at the current time instant. $-currentPR : D \times \{current\} \rightarrow (pr, \prec)$, is the priority of the command that the device is running at the current time instant. $(pr, \prec)$ is defined below. $-conflict : D \rightarrow 2^{DOA \times DOA} \cup \emptyset$, is a set of conflicting operation pairs defined for each device by the administrator, e.g. homeowner. $-conflict(d : D) = \{(op_i, op_j) \mid op_i \in DOA(d) \wedge op_j \in DOA(d)\}$. - All other functions are as defined in Table 5.1. $-(pr, \prec)$ is a totally ordered set of priorities with at least two distinct elements of $\perp$ and $\top$ which correspondingly represent the lowest and highest priorities in the system. $-prA : TE \rightarrow (pr, \prec)$ is a function which retrieves priority of the triggering events in the system, originally assigned by system administrator/homeowner. **Messages, Scenarios and Auxiliary Functions** $-A$, is a set of actions in the system. $-$For each action $a \in A : a = (s : D, m : M, r : D)$, action is defined as a triplet indicating communication of message *m* from device *s* to device *r*. $-S$ is the set of scenarios in the system defined by system's administrator/homeowner, which is defined below.

Continuation of Table 5.3

$-TeSA : TE \rightarrow 2^S$ is a one-to-many relation which defines a (set) of scenario(s) that would be provoked by a triggering event $te \in TE$.

−For each $s \in S : s =$

$\big(Action_s \subseteq A, tr_s : TE, pr_s : (pr, \prec), active : \{"true", "false"\}, id\big)$, and $pr_s = prA(tr_s)$.

$-active(s : S) = \begin{cases} "true" & \text{while } tr_s \text{ is in effect.} \\ "false" & \text{As soon as } tr_s, \text{ triggering event, reverts.} \end{cases}$

$-typeSet = \{"query", "command", "info"\}$ is a mandatory attribute of every message which indicates its *type* and thereby the rest of message attributes.

−For each $m \in M$, we assume the first attribute must determine the type of the message and the second attribute must determines its priority: $att_1 \in typeSet, att_2 \in (pr, \prec)$

$-typeSetAtt : M \rightarrow 2^{DAA} \cup 2^{DOA}$, is a function which indicates the set of attribute keys required to be communicated based on the message type, supposed to be communicated via $\{att_3, ..., att_n\}$ in each message.

$-msgPrA : D \times M \times D \rightarrow (pr, \prec)$ is a function which is checked each time a device s_d wants to create a command message m and assigns proper priority to it, in order to be sent to device r_d .

$-msgPrA(s : D, m : M, r : D) =$

$$\begin{cases} pr_s & \text{if } \Big(m.att_1 = "command"\Big) \wedge \Big(\exists s \in S : active(s) = "true"\Big) \wedge \\ & \Big(\exists a \in Action_s : a.s = s_d \wedge a.r = r_d \wedge typeSetAtt(a.m) = typeSetAtt(m)\Big) \\ \perp & \text{otherwise} \end{cases}$$

Chech Access Predicate

$-CheckAccess$ is evaluated when a sender device (s) wants to send a message (m) to a receiver device (r) in the context of current environment state ($current$) and is evaluated based on following formula:

$-CheckAccess(s : D, m : M, r : D, current : ES) \equiv$

$CheckAtt(s : D, m : M, r : D, current : ES) \wedge CheckPriority(s : D, m : M, r : D, current : ES) \wedge Authorization(s : D, m : M, r : D, current : ES)$

$$-CheckAtt = True \iff typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = "query" \\ \in DOA(r) & \text{if } m.value_1 = "command" \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = "info" \end{cases}$$

$-CheckPriority(s : D, m : M, r : D, current : ES) \equiv$

$$\begin{cases} "false" & \text{if } (m.att_1 = "command") \wedge \\ & \big[((m."op", currentOP(r)) \in conflict(r)) \wedge (m.value_2 \prec currentPR(r)\big] \\ "true" & \text{otherwise} \end{cases}$$

$-Authorization(s : D, m : M, r : D, current : ES)$ is a logical proposition which could be evaluated to either True or False and is created using following policy rules.

$-p \equiv (p) \mid \neg p \mid p \wedge p \mid p \vee p \mid \exists x \in set.p \mid \forall x \in set.p \mid set\ \Delta\ set \mid atomic \in set$

$-\Delta \equiv \subset | \subseteq | \not\subseteq | \cap | \cup$

End of Table

Messages, Scenarios and Auxiliary Functions. In formal representation of our model, A is the set of predefined actions, which basically includes a message and its sender and receiver. Each action

(a) is representing a message being communicated between two devices. S is a set of scenarios in the system. Each scenario, s, is representing a set of actions happening via messaging among IoT devices in the smart home. $TeSA$ is a function which defines a (set) of scenario(S) which would be triggered as the result of a given triggering event. As far as the triggering event is still effective, the triggered scenario(s) would also be going on and considered to be *active*. Priority of each scenario is the same as its trigger, and the priority value of messages included in one scenario would be the same as its containing scenario. Priorities are defined as binary relations between different triggers.

Check Access Predicate. The Check Access predicate includes three parts, each of which is responsible for a portion of access control in our extended model, and access would be granted to communicated the desired message if and only if all three following functions return TRUE:

1. $CheckAtt(s : D, m : M, r : D, current : ES)$ is a function which checks the feasibility of communication based on the message type and typeSetAtt in the message. It does the same checks as described in Section 5.2.2.

2. $CheckPriority(s : D, m : M, r : D, current : ES)$ is specific to the scenario-based model proposed in this section. As messages may conflict only if they are of type *command*, this function returns TRUE, for all messages of type *info* or *query*. For command messages, it returns FALSE when the requested operation via command message is in conflict with the current operation being done at the moment on the receiver or if the message to be sent is assigned with lower priority. Otherwise, it returns TRUE.

3. $Authorization(s : D, m : M, r : D, current : ES)$ is a logical proposition which determines the set of allowed communications in the smart home IoT environment between different devices.

5.3.3 Smart Home Use Case

Figure 5.4 represents a smart home use case for our scenario-based model. *Sender* and *receiver* fields in the messages are depicted in the figure, arrows represent the direction of communications.

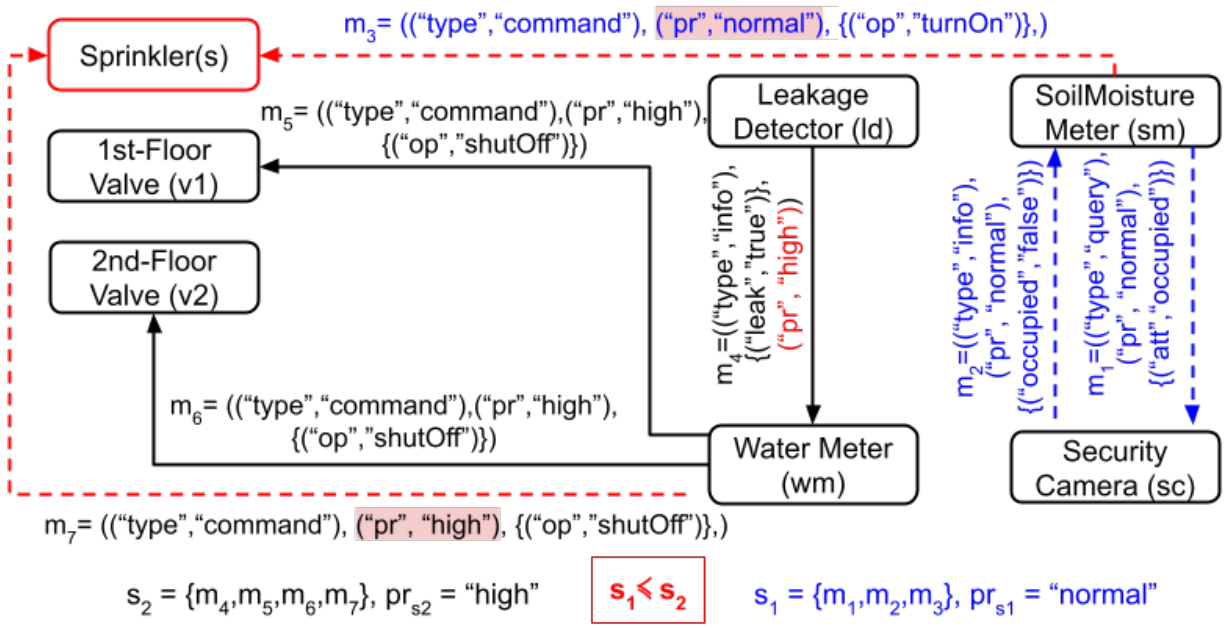

Figure 5.4: Smart Home Use Case for Scenario-Driven Device-to-Device Communication

This use case includes a two-story smart home IoT containing a soil moisture meter, an outdoor camera, a leakage detector, a sprinkler and two valves for first and second floor water flow control. A Soil moisture meter determines when the sprinkler starts spraying water by monitoring moisture level, and the leakage detector is responsible to cut off the water when any abnormal flow is discovered. We chose to exemplify scenario-based model by a different use case than previous one in Section 5.2.3, in which outdoor camera was responsible for all communications, however, herein different devices are messaging each other. Notably, none of the proposed models, not message-based, nor scenario based, need a central device to be responsible for all message communications.

Here, there are two scenarios going on. Scenario s_1 is initiated when the Soil Moisture Meter senses the soil moisture level to be lower than a specified threshold, so it sends a *query* message to the Security Camera in the backyard to inquire the outdoor's vacancy status, determined by *occupied* attribute value. If the backyard is vacant ($occupied = false$), then Soil Moisture Meter sends a *command* message with *normal* priority to *turn on* the Sprinkler. After a while when the sprinker is still spraying water, scenario s_2 is triggered when the Leakage Detector detects a leak, so it sends a *info* message to the Main Water Meter. Thereafter, Main Water Meter sends a

command message with *high* priority to first/second-floor Valves, and Sprinkler to shut the flow off. Here the message to shut off the Sprinkler would go through because the $currenPR$ is *normal* and the message from the Main Water Meter has higher priority. So, Sprinkler faces a conflict as it has received two conflicting commands from Soil Main Moisture Meter and Leakage Detector. In our model, Sprinkler executes whatever operation included in the *command* message with higher priority, which is shut off from Leakage Detector.

Table 5.4: Scenario-Based D2D Model: Smart Home Use Case

Begin of Table
Core Components D = {LeakageDetector, MainWaterMeter, SoilMoistureMeter, SecurityCamera, Sprinkler, FirstFloorValve, SecondFloorValve} OP = {StartRecording, StopRecording, StartMeasure, StopMeasure, StartMonitor, StopMonitor, TurnOn, ShutOff} ES = {current} DOA = {(SecurityCamera,{StartRecording, StopRecording}), (LeakageDetector,{StartMonitor, StopMonitor}), (MainWaterMeter, {StartMeasure, StopMeasure}), (SoilMoistureMeter, {StartMonitor, StopMonitor}), (Sprinkler, {TurnOn, ShutOff}), (FirstFloorValve, {TurnOn, ShutOff}), (SecondFloorValve, {TurnOn, ShutOff})} DAA = {(SecurityCamera,{id, type, location, recording, occupied}), (LeakageDetector,{id, type, leak}), (MainWaterMeter, {id, type, Measuring}), (SoilMoistureMeter, {id, type, droughtStatus}), (FirstFloorValve, {id, type, location, flowStatus}), (SecondFloorValve, {id, type, location, flowStatus}), (Sprinkler, {id, type, location, flowStatus})} EAA = {day, time} TE = {(LeakageDetector,(leak,"true")), (SoilMoistureMeter,(droughtStatus,"dry"))} $-Ent = D \cup ES \cup M \cup S \cup TE$
Attribute Functions $(pr, \prec) = (\bot \prec low \prec normal \prec high \prec \top)$ id(SecurityCamera) = "sc", type(SecurityCamera) = "cameras", location(SecurityCamera) = "outdoor", recording(SecurityCamera) = {"true","false"}, occupied(SecurityCamera) = {"true","false"}, conflict(SecurityCamera)={(StartRecording,StopRecording)} id(LeakageDetector) = "ld", type(LeakageDetector)= "detectors", subtype(LeakageDetector)="leak", leak(LeakageDetector) = {"true","false"}, conflict(LeakageDetector)={(StartMonitor, StopMonitor)} id(MainWaterMeter) = "wm", type(MainWaterMeter)= "valves", subtype(MainWaterMeter)="watermeter",Measuring(MainWaterMeter) = {"true", "false"}, type(MainWaterMeter)= "valves", conflict(MainWaterMeter)={(StartMeasure, StopMeasure)}

Continuation of Table 5.4

id(SoilMoistureMeter) = "sm", type(SoilMoistureMeter)= "detectors", droughtStatus(SoilMoistureMeter) = {"dry","moist"}, conflict(SoilMostureMeter)={(StartMonitor, StopMonitor)}

id(FirstFloorValve) = "v1", type(FirstFloorValve)="valves", location(FirstFloorValve)="indoor", flowStatus(FirstFloorValve) = {"flowing", "halted"}, conflict(FirstFloorValve)={(TurnOn, ShutOff)}
id(SecondFloorValve) = "v2", conflict(SecondFloorValve)={(TurnOn, ShutOff)}, flowStatus(SecondFloorValve) = {"flowing", "halted"},

id(SecondFloorValve) = "v1", type(SecondFloorValve)="valves", location(SecondFloorValve)="indoor", flowStatus(SecondFloorValve) = {"flowing", "halted"}, conflict(SecondFloorValve)={(TurnOn, ShutOff)}

id(Sprinkler) = "s", type(Sprinkler)="valves", location(Sprinkler)="outdoor", flowStatus(Sprinkler) = {"flowing", "halted"}, conflict(Sprinkler)={(TurnOn, ShutOff)}

$-attAssignType(d, att) = \begin{cases} static & \text{if } att \in \{"id", "type", "subtype", "location"\} \\ dynamic & \text{otherwise} \end{cases}$

Message, Scenarios and Auxiliary Functions

$-(pr, \prec) = (\bot \prec low \prec normal \prec high \prec \top)$
$-prA = \{((LeakageDetector, leak = "true"), high),$
$((SoilMoistureMeter, droughtStatus = "dry"), normal)\}$
$-A = a_1, a_2, a_3, a_4, a_5, a_6, a_7$
$-a_1 = (SoilMoistureMeter, \{("type", "query"), ("att", "occupied")\}, SecurityCamera)$
$-a_2 = (SecurityCamera, \{("type", "info"), ("occupied", "false")\}, SoilMoistureMeter)$
$-a_3 = (SoilMoistureMeter, \{("type", "command"), ("op", "TurnOn")\}, SecurityCamera)$
$-a_4 = (LeakageDetector, \{("type", "info"), ("leak", "true")\}, MainWaterMeter)$
$-a_5 = (MainWaterMeter, \{("type", "command"), ("op", "ShutOff")\}, FirstFloorValve)$
$-a_6 = (MainWaterMeter, \{("type", "command"), ("op", "ShutOff")\}, SecondFloorValve)$
$-a_7 = (MainWaterMeter, \{("type", "command"), ("op", "ShutOff")\}, Sprinkler)$
$-S = \{s_1, s_2\}$
$-s_1 = \{Action_{s_1}, pr_{s_1}, tr_{s_1}, active, id\}$
$-Action_{s_1} = \{a_1, a_2, a_3\}, tr_{s_1} = (SoilMoistureMeter, (droughtStatus, "dry"), pr_{s_1} = "normal"$
$-s_2 = \{Action_{s_2}, pr_{s_2}, tr_{s_2}, active, id\}$
$-Action_{s_2} = \{a_4, a_5, a_6, a_7\}, tr_{s_2} = (LeakageDetector, (leak, "true"), pr_{s_2} = "high"$
$-id(s_1) = s, id(s_2) = s'$
$-active(s_1)$ and $active(s_2)$ is determined based on definition in Table 5.3, at each instant of time.
$-M = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$
$m_1 = (("type", "query"), ("pr", msgPrA(sender_1, m_1, receiver_1)), \{("att", "occupied")"\})$
$m_2 = (("type", "info"), ("pr", msgPrA(sender_2, m_1, receiver_2)), \{("occupied", "false"))\}$
$m_3 = (("type", "command"), ("pr", msgPrA(sender_3, m_3, receiver_3)), \{("op", "turnOn"))\}$
$m_4 = (("type", "info"), ("pr", msgPrA(sender_4, m_4, receiver_4)), \{("leak", "true")"\})$
$m_5 = (("type", "command"), ("pr", msgPrA(sender_5, m_5, receiver_5)), \{("op", "shutOff")"\})$
$m_6 = (("type", "command"), ("pr", msgPrA(sender_6, m_6, receiver_6)), \{("op", "shutOff")"\})$
$m_7 = (("type", "command"), ("pr", msgPrA(sender_7, m_7, receiver_7)), \{("op", "shutOff")"\})$

Continuation of Table 5.4

-After s_1's activation: $\begin{cases} msgPrA(m_1) = msgPrA(m_2) = \bot \\ msgPrA(m_3) = "normal" \end{cases}$

-After s_2's activation: $\begin{cases} msgPrA(m_4) = \bot \\ msgPrA(m_5) = msgPrA(m_6) = msgPrA(m_7) = "high" \end{cases}$

Attribute Authorization Function

$-CheckAccess(s : D, m : M, r : D, current : ES) \equiv$
$CheckAtt(s : D, m : M, r : D, current : ES) \wedge CheckPriority(s : D, m : M, r : D, current : ES) \wedge$
$Authorization(s : D, m : M, r : D, current : ES)$

$-CheckAtt(s : D, m : M, r : D, current : ES) = True \iff$

$$typeSetAtt(m) = \begin{cases} \subseteq 2^{DAA(r)} & \text{if } m.value_1 = "query" \\ \in DOA(r) & \text{if } m.value_1 = "command" \\ \subseteq 2^{DAA(s)} & \text{if } m.value_1 = "info" \end{cases}$$

$-CheckPriority(SoilMoistureMeter, m_1, SecurityCamera, \{current\}) \equiv "true"$
$-CheckPriority(SecurityCamera, m_2, SoilMoistureMeter, \{current\}) \equiv "true"$
$-CheckPriority(SoilMoistureMeter, m_3, Sprinkler, \{current\}) \equiv "true"$
$-CheckPriority(LeakageDetector, m_4, MainWaterMeter, \{current\}) \equiv "true"$
$-CheckPriority(MainWaterMeter, m_5, FirstFloorValve, \{current\}) \equiv "true"$
$-CheckPriority(MainWaterMeter, m_6, SecondFloorValve, \{current\}) \equiv "true"$
$-CheckPriority(MainWaterMeter, m_7, Sprinkler, \{current\}) \equiv "true"$

$-Authorization(s : D, m : M, r : D, current : ES) \equiv q_1 \vee q_2 \vee q_3 \vee q_4 \vee q_5$

$$q_1 = \Big[\Big(m.att_1 = "info"\Big) \wedge \Big(typeSetAtt(m) \in \{"leak"\}\Big) \wedge \Big(type(s) = "detectors"\Big) \wedge \Big(type(r) = "valves"\Big) \wedge \Big(subtype(r) = "watermeter"\Big)\Big]$$

$$q_2 = \Big[\Big(m.att_1 = "command"\Big) \wedge \Big(typeSetAtt(m) \in \{"ShutOff, TurnOn"\}\Big) \wedge \Big(type(r) = type(s) = "valves"\Big) \wedge \Big(subtype(s) = "watermeter"\Big)\Big]$$

$$q_3 = \Big[\Big(m.att_1 = "query"\Big) \wedge \Big(typeSetAtt(m) \in \{"occupied"\}\Big) \wedge \Big(type(s) = "detectors"\Big) \wedge \Big(subtype(s) = "soil"\Big) \wedge \Big(type(r) = "cameras"\Big) \wedge \Big(location(r) = "outdoor"\Big)\Big]$$

$$q_4 = \Big[\Big(m.att_1 = "info"\Big) \wedge \Big(typeSetAtt(m) \in \{"occupied"\}\Big) \wedge \Big(type(s) = "camera"\Big) \wedge \Big(location(source) = "outdoor"\Big) \wedge \Big(type(r) = "detectors"\Big) \wedge \Big(subtype(r) = "soil"\Big)\Big]$$

Continuation of Table 5.4
$q_5 = \Big(m.att_1 = "command"\Big) \wedge \Big(typeSetAtt(m) \in \{"TurnOn, ShutOff"\}\Big) \wedge \Big(type(s) = "detectors"\Big) \wedge \Big(subtype(s) = "soil"\Big) \wedge \Big(type(r) = "valves"\Big) \wedge \Big(location(r) = "outdoor"\Big)$
End of Table

Our proposed access control could be configured as presented in Table 5.4 to achieve following goals:

1. Authorize Soil Moisture Meter and Leakage detector to initiate appropriate set of actions by activating above-mentioned scenarios.

2. Enabling any device which receives conflicting *command* messages to resolve the conflict relying on priorities.

The smart home IoT devices, their attributes and available operations, and environment attribute and state are represented as *core components* of our model. The set of attributes for each device are represented under *attribute functions*. Five different priorities have been defined as a binary relation between scenarios and their contained messages. There are two different scenarios, s_1 and s_2, each of which is a set of actions. The order of communicated messages in each scenario is indicated by their subscript number. Based on assigned priorities to triggers, which are $(LeakageDetector, (leak, true))$ with *high* priority and $(SoilMoistureMeter, droughtStatus = dry))$ with *normal* priority, indicated by prA function. The two scenarios of s_1 and s_2 would also have the same prioritites astheir triggering events and are comparable as $s_1 \prec s_2$. Check Access predicates have also been presented in Table 5.4, which include policy rules for CheckAtt, CheckPriority and Authorization of message communication between IoT device pairs.

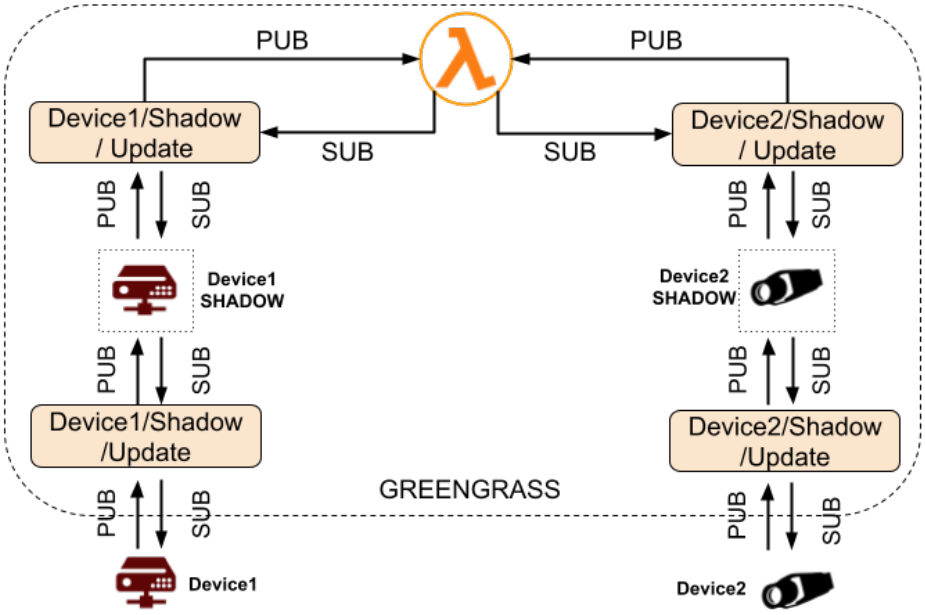

Figure 5.5: Device-to-Device Architecture

5.4 Enforcement Architecture

The Internet Architecture Board (IAB) has released an informational RFC [200] considering heterogeneity of IoT devices which desired to directly interoperate and communicate. Based on this, Beltran et al. proposed five IoT communication models and identified three architectures that embody decentralized authorization of devices [34]. However, direct device-to-device communications is considered to be a long haul. In this section, we propose a decentralized enforcement architecture for our model, in which different IoT devices delegate the authorization decisions to an external entity. Even if heterogeneous IoT devices could communicate directly to each other, reliance upon an external entity to embrace decentralized access control is still desirable, as restricted IoT devices would no longer need to maintain/apply authorization information/rules themselves [34].

Our enforcement architecture is depicted in Figure 5.5, which is designed based on AWS IoT. However, our enforcement architecture is not specific to AWS and could be designed independently of any cloud provider. We use AWS because of its simplicity, security and flexibility and being

agnostic to device type and OS [4]. AWS could be used to manage a variety of devices ranging from micro-controllers to connected cars, as it is agnostic to the device type and OS. AWS IoT Device Management is agnostic to device type and OS [5]. We deployed our architecture utilizing AWS Greengrass SDK [6].

AWS Greengrass (GG) is an edge run-time which seamlessly extends AWS to physical devices, so IoT devices would be able to communicate with each other through local messaging. It also acts as an intermediary for communications with the cloud, even for not IP enabled devices, or while network connection is intermittent. Greengrass utilization enables devices to autonomously react to local events and securely communicate with each other over the local network. Each device has a corresponding shadow (virtual device) on Greengrass, which persistently keeps the latest known state of the device. Device shadows remain accessible all the time, even if the device itself is not connected to the cloud. Devices would interact over MQTT [7] protocol in AWS, which is a lightweight machine-to-machine publish-subscribe protocol designed for constrained devices. Using MQTT, devices will communicate on their private topics, *device/shadow/update*, to update their shadows, and trigger the local computation function, known as *Lambda* function(λ) [8].

As devices communicate their state on their private topic, it would be stored locally, which is known as the device shadow and is accessible only by themselves and the lambda function. Authorization on AWS is policy-based. We defined access control in the *policy.json* file. Authorization rules in the policy file have been defined based on the status of communicating IoT devices (devices' attributes), environmental context, and the contexts of communicated messages. When Lambda code is triggered, it executes the code we developed to check the policy. If the policy rules grant permission, the results will be communicated, the shadow states will be updated, and the physical device will respond appropriately.

[4]https://aws.amazon.com/iot-device-management/
[5]https://aws.amazon.com/iot-device-management/
[6]https://docs.aws.amazon.com/greengrass/
[7]http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html
[8]https://aws.amazon.com/lambda

Table 5.5: Full Experiment and Device State Update Statistics.

Expr.	Min	10%	25%	Median	75%	90%	95%	99%	Max
Full Timer	24.19	25.36	26.04	34.92	36.70	39.09	42.93	47.36	67.20
State Update	3.50	3.61	3.73	5.71	6.61	6.90	7.37	10.21	21.85

5.5 Implementation

5.5.1 Experiment Setup

As a proof-of-concept, we simulated two smart home use cases discussed in Sections 5.2.3 and 5.3.3. Our use cases implemented using AWS IoT Greengrass v1, running on a dedicated virtual machine with one virtual CPU, 2 GB of RAM and 20 GB hard drive. The operating system of the virtual machine is Ubuntu 20.4.2 LTS and it is connected to a 1 Gbps network. We wrote the AWS lambda code on Greengrass in Python 3.8 and it is running in a long-lived isolated docker environment with limited RAM of 64 MB. Any message communication between two IoT devices would wake up the lambda function as it is reflected in the device shadows on Greengrass and would be published to lambda. Then, lambda checks access control policy which has been contained in *policy.json* based on the contents of the message, reflecting the attributes and attributes of device shadows, i.e. devices' attributes, to govern the requested communication. Any changes in devices' attributes as a result of authorization decisions would be reflected in device shadows, according to what has been defined in the IMPACT function in our model.

5.5.2 Implementation Results

Both scenarios we implemented have a similar computational process. A device would send a message to another device through the GG lambda function. Upon receiving the message, lambda spins up multiple threads, loads the necessary Python libraries, and begins processing our code. This initial process results in a few spurious results that have particularly long processing delays

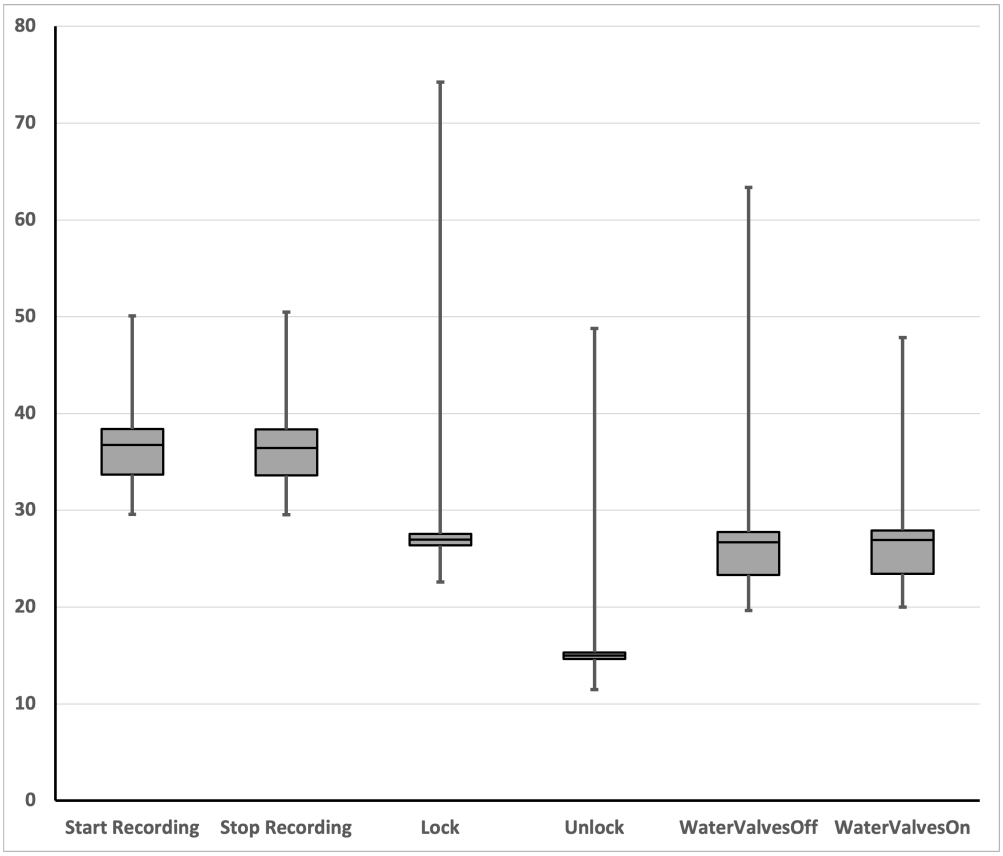

Figure 5.6: Time per Action on Greengrass

shown at the top 1% of results depicted in Table 5.5. As a message received by lambda, once the sender and destination devices are authenticated, the code would proceed. In general, lambda would inquire the shadow status of the device(s) as necessary, the priority levels of messages, then executes a series of commands to various devices based on the policy.json. Any priority conflicts between *command* messages received by the same device is resolved based on this rule: the higher priority is dominant. In terms of two conflicting commands with the same priorities, we run the most recent command. If the scenario did not exist in the policy, the situation would be considered invalid, disregarded, and no additional commands would be executed.

Our implementation results show an average time for a device to execute an action using our model/architecture, have lambda process it, and update the respective devices is, on average, 35 milliseconds. This complete process is indicated as 'full timer' and is an average over 500 trials and includes transmission time going to/from lambda to the devices over a 1 Gbps network connection.

A more detailed breakdown of the analysis is included in Table 5.5. The time to update a device's state, for example a door lock going from unlocked to locked, is on average 6ms. The various use cases and actions are also timed and shown as boxplots in Figure 5.6, where the whiskers are the minimum and maximum values collected across 500 trials for each experiment. The box shows the 25 percentile and 75 percentile of the data, with median as the line contained inside the box.

5.6 Discussion: Model/Architecture Properties and Limitations

The proposed model in this research is unparalleled as it provides specification for mediating access in device-to-device communications for the first time. Besides being context-aware, dynamic and lightweight, our model also provides following features.

Continuity of Access. Our message-based access control paradigm relies on attributes of sender, receiver, environment, and the (key,value) attribute pairs communicated via messaging. However, attributes of sender/receiver devices may change at any time as a result of an operation or an alteration in its environment, a.k.a mutability [171]. It is required for an access control model designed for dynamic IoT environments to maintain regular surveillance over attribute values of participating devices and continuously evaluate the policy based on that. We utilized AWS for our model enforcement, in which lambda would be informed on any changes in participating device attributes via subscription to corresponding topics. Therefore, in case of any changes in attributes of device or environment topics, lambda would be triggered, re-evaluate the policy and adjust the access authorizations accordingly. For example, in use case discussed in Section 5.3.3, if the outdoor camera detects the owner returns home, it send a *finish* message to door lock and cameras for unlocking the door and stop recording. Here, the continuity is supported via our proposed enforcement architecture. In order to provide continuity as one of the model's elements, regardless of its enforcement method, the continuous retrieval and evaluation of entity/environment attributes should be incorporated in the model. Compatible with requirements of modern IoT environments, UCON [171] would be well suited to equip the access control model to promptly react to attribute

changes while the previously granted access is being utilized. Moreover, continuity of access control could be concluded when the proposed model is able to revoke the previously granted access in case of any unintended change in attributes. We justify our proposal still lacks access continuity at model level. There are some research works trying to add continuity at application level in IoT environments by integration of MQTT and UCON [128].

Architecture Agnostic. AWS as the enforcement architecture augmented our framework with some desired features, however, our model is not peculiar to it.

Chapter 6: CONCLUSION AND FUTURE WORK

This chapter provides a summary of contributions in this dissertation and some open research problems which potentially could be further investigated.

6.1 Summary

This dissertation fundamentally contributes to the operational and administrative access control in smart home IoT environments. We conduce to improve safety and security in this rapidly evolving application of IoT by providing access control-related solutions which affect or directly provide authorization in a smart home.

Our first contribution addresses the *inconsistency* problem in smart homes in which intermittent internet connections may cause access violation resulting from exposure of decision point to outdated information. We formulated different levels of safety and consistency as formal specifications for any distributed attribute-based access control environment in general, along with smart home IoT interpretations. Moreover, we introduced the *refresh* concept instead of old revocation check, for the first time, which provides new attribute values rather than simply invalidating old ones. Utilizing this approach augments these environments with more safety and availability. We also realize the safety and consistency may be compromised due to concurrent usage of mutable attributes. To tackle that, we presented two quota-based categories of practical scenarios including user-based and service based, exemplified in a smart home environment.

Second, we recognize an access control solution to be incomplete if administration is not handled. We undertake the need to develop administrative models in order to govern access changes in a dynamically changing environment such as smart home IoT. So, the ever-changing nature of operational models has to be considered in order to efficiently perform administration. We proposed a role-based access control administrative model based on EGRBAC, which was the operational model of our choice for the smart home IoT. We realize administration to be best done if decentralized, which helps to manage the single point of failure as well as improving users' privacy

by introducing administrative units in our solution. We outlined the formal specification of the proposed model and consolidated the presented ideas by proposing smart home case studies.

Furthermore, we presented a decentralized, ledger-based, publish-subscribe based architecture for the administration of access in a smart home IoT environment to preside at the assignments of underlying operational authorizations. Proposed architecture is endorsed by a proof-of-concept implementation on top of Ethereum official testnet, i.e. Ropsten. We utilized smart contracts to ensure the integrity of administration supplemented by intrinsic benefits of blockchain to be distributed and transparent. We recognize blockchain could bring its intrinsic advantages of distribution, transparency, and scalability to the administration of access, while it is not yet practical to be used for operational access control which is reassured based on our implementation results. This assertion sheds light on the hype around utilizing blockchain at the operational level of access control, whereas administrative solutions are shown to successfully utilize blockchain benefits.

Uttermost, we proposed a novel device-to-device access control solution for the smart home IoT environment. Despite the extensive research on enabling technologies for interoperability of heterogeneous IoT devices, designing appropriate access control models is scarcely investigated. Our approach defines a set of authorized message flows between devices through establishing access control rules, by unprecedentedly utilizing the message passing paradigm for access control. Moreover, we introduced the concept of scenarios, reflecting a chain of actions in the smart home initiated by a triggering event. We defined a total order relation based on priorities among triggers, thereby scenarios and their contained messages. So, we can handle the probable conflicts when the same device receives conflicting commands or simply drop the messages with lower priority. Viability of our approach is substantiated via a formal model and an enforcement architecture, which is also backed up by a proof-of-concept implementation.

6.2 Future Work

There are several potential directions of research which could be explored as extensions to the research presented in this dissertation, as follows:

1. No access control framework considered as a complete solution, without including both operational and administrative approaches. Developing administrative solutions for device-to-device interoperability is a direction which could be further investigated. In this context, proposed approach for device-to-device access control, could be coupled with an administrative model to configure the legitimate D2D communications, and define corresponding access rules.

2. Building access control solutions which are tailored to specific features of IoT devices and environments is an utmost requirement. Another research direction is to propose a comprehensive IoT-specific authorization solution at the operational level of access control, which entails both user-to-device and device-to-device access mediation.

3. To cope with authorization requirements in dynamic IoT environments, utilizing access control approaches which consider mutability of attributes of subjects and objects in an IoT application environment is of utmost importance to be considered. Moreover, monitoring the access after being granted, a.k.a continuity of access control is required for most IoT applications. It is also desirable for any access control model to be able to revoke the previously granted access in case of any unintended change in attributes. Intrinsic consideration of mutability and continuity of access at model-level, is to be sought-after, so the approach would be agnostic to any specific architecture or implementation technology.

4. A factual interoperability solution for IoT environments would be obtained when there is no need to rely on a gateway for communication of heterogeneous IoT devices. To enable device-to-device interoperability, it is required for communication of two or more IoT platforms to be facilitated.

5. Although there is a single administrative role assigned for each administrative unit/sub-unit in our model in this dissertation, dissension among different administrator users in that role is possible. So, the permission granted by one administrator user may be revoked shortly after by another administrator user. Moreover, conflicts may happen due to different devices

being shared in a smart home IoT among different home users. Therefore, it is required to incorporate a conflict resolution policy in the model.

BIBLIOGRAPHY

[1]

[2] Ethereum evm illustrated. Available at https://github.com/takenobu-hs/ethereum-evm-illustrated.

[3] Saving the future of the Internet of Things, 2015. Available at https://www.ibm.com/downloads/cas/Y5ONA8EV.

[4] Amazon GreenGrass, 2021. Available at https://docs.aws.amazon.com/greengrass/.

[5] Chrony, 2021. Available at https://chrony.tuxfamily.org/.

[6] Device authentication and authorization for AWS IoT Greengrass, 2021. Available at https://docs.aws.amazon.com/greengrass/v1/developerguide/device-auth.html.

[7] Ethereum 2.0 proof of stake, 2021. Available at https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

[8] ETHEREUM: A secure decentralised generalised transaction ledger, EIP-150 REVISION, 2021. Available at https://gavwood.com/paper.pdf.

[9] Ethereum Average Block Time Chart, 2021. Available at https://etherscan.io/chart/blocktime.

[10] Ethereum races clock to collect enough coins for big upgrade, 2021. Available at https://www.bloomberg.com/news/articles/2020-11-23/ethereum-races-clock-to-collect-enough-coins-for-huge-upgrade.

[11] Ethereumnetworks, 2021. Available at https://ethereum.org/en/developers/docs/networks/.

[12] Hype cycle for emerging technologies, 2021. Available at https://www.gartner.com/smarterwithgartner/3-themes-surface-in-the-2021-hype-cycle-for-emerging-technologies.

[13] Infura, 2021. Available at https://infura.io/product.

[14] OWASP IoT Project, 2021. Available at https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10.

[15] Remix IDE, 2021. Available at https://remix.ethereum.org/.

[16] Security best practices for AWS IoT Greengrass, 2021. Available at https://docs.aws.amazon.com/greengrass/v1/developerguide/security-best-practices.html.

[17] Statista: Smart Home Market, 2021. Available at https://www.statista.com/outlook/dmo/smart-home/united-states.

[18] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing (HUC)*. Springer.

[19] Atul Adya. *Weak consistency: a generalized theory and optimistic implementations for distributed transactions*. PhD thesis, MIT, 1999.

[20] Gail-Joon Ahn, Hongxin Hu, and Jing Jin. Towards role-based authorization for OSGi service environments. In *Future Trends of Distributed Computing Systems*. IEEE, 2008.

[21] The OSGi Alliance. OSGi service platform core specification.

[22] Mousa Alramadhan and Kewei Sha. An overview of access control mechanisms for internet of things. In *International Conference on Computer Communication and Networks (ICCCN)*.

[23] Asma Alshehri and Ravi Sandhu. Access control models for cloud-enabled internet of things: A proposed architecture and research agenda. In *Collaboration and Internet Computing (CIC)*. IEEE, 2016.

[24] Safwa Ameer, James Benson, and Ravi Sandhu. The EGRBAC model for smart home IoT. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 457–462. IEEE, 2020.

[25] Safwa Ameer, James Benson, and Ravi Sandhu. An attribute-based approach toward a secured smart-home iot access control and a comparison with a role-based approach. *Information*, 2022.

[26] Safwa Ameer and Ravi Sandhu. The HABAC model for smart home iot and comparison to EGRBAC. In *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, pages 39–48, 2021.

[27] Mahmoud Ammar, Giovanni Russello, and Bruno Crispo. Internet of things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications*, 2018.

[28] Bayu Anggorojati, Parikshit Narendra Mahalle, Neeli Rashmi Prasad, and Ramjee Prasad. Capability-based access control delegation model on the federated IoT network. In *International Symposium on Wireless Personal Multimedia Communications*. IEEE, 2012.

[29] Orlando Arias, Jacob Wurm, Khoa Hoang, and Yier Jin. Privacy and security in internet of things and wearable devices. *Transactions on Multi-Scale Computing Systems*, 2015.

[30] Bröring Arne, Zappa Achille, Vermesan Ovidiu, Främling Kary, Zaslavsky Arkady, Gonzalez-Usach Regel, Szmeja Pawel, Carlos E Palau, Jacoby Michael, Ivana Podnar Zarko, et al. *Advancing IoT platforms interoperability*. 2018.

[31] Fatih Bakir, R Wolski, and C Krintz. Caplets: Resource aware capability-based access control for IoT. In *Symposium on Edge Computing (SEC)*. IEEE, 2021.

[32] Shanay Behrad, Emmanuel Bertin, Stéphane Tuffin, and Noel Crespi. A new scalable authentication and access control mechanism for 5G-based IoT. *Future Generation Computer Systems*, 2020.

[33] Oladayo Bello and Sherali Zeadally. Intelligent device-to-device communication in the internet of things. *Systems Journal*, 2014.

[34] Victoria Beltran and Antonio F Skarmeta. Overview of device access control in the IoT and its challenges. *Communications Magazine*, 2018.

[35] Jorge Bernal Bernabe, Jose Luis Hernandez Ramos, and Antonio F Skarmeta Gomez. TACIoT: multidimensional trust-aware access control system for the Internet of Things. *Soft Computing*, 20(5):1763–1779, 2016.

[36] Philip A Bernstein and Nathan Goodman. Concurrency control in distributed database systems. In *ACM Computing Surveys*, 1981.

[37] Emmanuel Bertin, Dina Hussein, Cigdem Sengul, and Vincent Frey. Access control in the internet of things: A survey of existing approaches and open research questions. *Annals of telecommunications*, 2019.

[38] Bruhadeshwar Bezawada, Kyle Haefner, and Indrakshi Ray. Securing home iot environments with attribute-based access control. In *Workshop on Attribute-Based Access Control*. ACM, 2018.

[39] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. Access control model for AWS internet of things. In *International Conference on Network and System Security*. Springer, 2017.

[40] Smriti Bhatt and Ravi Sandhu. ABAC-CC: Attribute-based access control and communication control for internet of things. In *Symposium on Access Control Models and Technologies (SACMAT)*, 2020.

[41] Leepakshi Bindra, Changyuan Lin, et al. Decentralized access control for smart buildings using metadata and smart contracts. In *International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*. IEEE, 2019.

[42] Dario Bonino, Emiliano Castellina, and Fulvio Corno. The dog gateway: enabling ontology-based intelligent domotic environments. *Transactions on consumer electronics*, 2008.

[43] Arne Bröring, Stefan Schmid, Corina-Kim Schindhelm, Abdelmajid Khelil, Sebastian Käbisch, Denis Kramer, Danh Le Phuoc, Jelena Mitic, Darko Anicic, and Ernest Teniente. Enabling IoT ecosystems through platform interoperability. *IEEE software*, 34(1):54–61, 2017.

[44] Arne Bröring, Andreas Ziller, Victor Charpenay, Aparna S Thuluva, Darko Anicic, Stefan Schmid, Achille Zappa, Mari Paz Linares, Lars Mikkelsen, and Christian Seidel. The big iot api-semantically enabling IoT interoperability. *Pervasive Computing*, 2018.

[45] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[46] Jan Camenisch and Els Van Herreweghen. Design and implementation of the Idemix anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30. ACM, 2002.

[47] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. Sensitive information tracking in commodity {IoT}. In *USENIX Security Symposium*, 2018.

[48] Melissa Chase. Multi-authority attribute based encryption. In *Theory of Cryptography Conference*. Springer, 2007.

[49] Melissa Chase and Sherman SM Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM CCS*, 2009.

[50] Manuel Cheminod, Luca Durante, Fulvio Valenza, and Adriano Valenzano. Toward attribute-based access control policy in industrial networked systems. In *International Workshop on Factory Communication Systems (WFCS)*, pages 1–9. IEEE, 2018.

[51] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, 2020.

[52] Andrea Cimmino, Viktor Oravec, Fernando Serena, Peter Kostelnik, María Poveda-Villalón, Athanasios Tryferidis, Raúl García-Castro, Stefan Vanya, Dimitrios Tzovaras, and Christoph Grimm. VICINITY: IoT semantic interoperability based on the web of things. In *Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019.

[53] Pietro Colombo and Elena Ferrari. Access control enforcement within MQTT-based internet of things ecosystems. In *Symposium on Access Control Models and Technologies(SACMAT)*. ACM, 2018.

[54] Jason Crampton and George Loizou. Administrative scope: A foundation for role-based administrative models. *Transactions on Information and System Security (TISSEC)*, 2003.

[55] Jason Paul Cruz, Yuichi Kaji, and Naoto Yanai. RBAC-SC: Role-based access control using smart contract. *IEEE Access*, 2018.

[56] Luis Cruz-Piris, Diego Rivera, Ivan Marsa-Maestre, Enrique De La Hoz, and Juan R Velasco. Access control mechanism for iot environments based on modelling communication procedures as resources. *Sensors*, 2018.

[57] MAC Dekker et al. Rbac administration in distributed systems. In *Symposium on Access Control Models and Technologies*. ACM, 2008.

[58] Tamara Denning, Tadayoshi Kohno, and Henry M Levy. Computer security and the modern home. *Communications Journal*, 2013.

[59] David Derler, Kai Samelin, Daniel Slamanig, and Christoph Striecks. Fine-grained and controlled rewriting in blockchains: Chameleon-Hashing gone attribute-based. *IACR Cryptol.*, 2019.

[60] Monika Di Angelo and Gernot Salzer. A survey of tools for analyzing Ethereum smart contracts. In *Decentralized Applications and Infrastructures*. IEEE, 2019.

[61] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[62] Danny Dolev and Andrew Yao. On the security of public key protocols. *Transactions on information theory*, 1983.

[63] Yuji Dong, Kaiyu Wan, Xin Huang, and Yong Yue. Contexts-states-aware access control for Internet of Things. In *Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2018.

[64] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for IoT security and privacy: The case study of a smart home. In *International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017.

[65] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. LSB: A lightweight scalable blockchain for iot security and anonymity. *Journal of Parallel and Distributed Computing*, 2019.

[66] Sofia Dutta, Sai Sree Laya Chukkapalli, et al. Context sensitive access control in smart home environments. In *Big Data Security on Cloud*. IEEE, 2020.

[67] Sofia Dutta, Sai Sree Laya Chukkapalli, Madhura Sulgekar, Swathi Krithivasan, Prajit Kumar Das, and Anupam Joshi. Context sensitive access control in smart home environments. In *BigDataSecurity, HPSC and IDS*. IEEE, 2020.

[68] Sofia Dutta, Sai Sree Laya Chukkapalli, Madhura Sulgekar, Swathi Krithivasan, Prajit Kumar Das, and Anupam Joshi. Context sensitive access control in smart home environments. In *Big Data Security on Cloud (BigDataSecurity), High Performance and Smart Computing,(HPSC) and Intelligent Data and Security (IDS)*. IEEE, 2020.

[69] Sofia Dutta, Sai Sree Laya Chukkapalli, Madhura Sulgekar, Swathi Krithivasan, Prajit Kumar Das, Anupam Joshi, et al. Context sensitive access control in smart home environments. In *Big Data Security on Cloud (BigDataSecurity 2020)*. IEEE, 2020.

[70] European Platforms Initiatives (EPI). Adaptive Gateways for dIverse muLtiple Environments.

[71] European Platforms Initiatives (EPI). Open virtual neighbourhood network to connect IoT infrastructures and smart objects.

[72] Jingwen Fan, Yi He, Bo Tang, Qi Li, and Ravi Sandhu. Ruledger: Ensuring execution integrity in trigger-action IoT platforms. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[73] Alexander Felfernig, Seda Polat Erdeniz, Paolo Azzoni, Michael Jeran, Arda Akcay, and Charalampos Doukas. Towards configuration technologies for IoT gateways. In *International Configuration Workshop*, 2016.

[74] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *Symposium on security and privacy (S&P)*. IEEE, 2016.

[75] David Ferraiolo, Ramaswamy Chandramouli, Rick Kuhn, and Vincent Hu. Extensible access control markup language (XACML) and next generation access control (NGAC). In *Workshop on Attribute Based Access Control*. ACM, 2016.

[76] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *Transactions on Information and System Security (TISSEC)*, 2001.

[77] Giancarlo Fortino, Claudio Savaglio, Carlos E Palau, Jara Suarez de Puga, Maria Ganzha, Marcin Paprzycki, Miguel Montesinos, Antonio Liotta, and Miguel Llop. Towards multi-layer interoperability of heterogeneous IoT platforms: The INTER-IoT approach. 2018.

[78] Behrang Fouladi and Sahand Ghanoun. Honey, iâm home!!, hacking zwave home automation systems. *Black Hat USA*, 2013.

[79] Behrang Fouladi and Sahand Ghanoun. Honey, Iâm home!!, hacking zwave home automation systems. *Black Hat USA*, 2013.

[80] Mario Frustaci, Pasquale Pace, and Gianluca Aloi. Securing the IoT world: Issues and perspectives. In *Standards for Communications and Networking (CSCN)*. IEEE, 2017.

[81] William C Garrison et al. On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In *IEEE S&P*, 2016.

[82] Dimitris Geneiatakis, Ioannis Kounelis, Ricardo Neisse, Igor Nai-Fovino, Gary Steri, and Gianmarco Baldini. Security and privacy issues for an IoT based smart home. In *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017.

[83] Ivan Gojmerac, Peter Reichl, Ivana Podnar Žarko, and Sergios Soursos. Bridging IoT islands: the symbIoTe project. *e & i Elektrotechnik und Informationstechnik*, 2016.

[84] Rohit Goyal, Nicola Dragoni, and Angelo Spognardi. Mind the tracker you wear: a security analysis of wearable health trackers. In *Symposium on Applied Computing*. ACM, 2016.

[85] Yajuan Guan, Juan C Vasquez, Josep M Guerrero, Natalie Samovich, Stefan Vanya, Viktor Oravec, Raúl García-Castro, Fernando Serena, María Poveda-Villalón, Carna Radojicic, et al. An open virtual neighbourhood network to connect IoT infrastructures and smart objectsâvicinity: Iot enables interoperability as a service. In *Global Internet of Things Summit (GIoTS)*. IEEE, 2017.

[86] Khalida Guesmia and Narhimene Boustia. OrBAC from access control model to access usage model. *Applied Intelligence*, 2018.

[87] Hao Guo, Ehsan Meamari, and Chien-Chung Shen. Multi-authority attribute-based access control with smart contract. In *International Conference on Blockchain Technology*. ACM, 2019.

[88] Rachael Harding, Dana Van Aken, Andrew Pavlo, and Michael Stonebraker. An evaluation of distributed concurrency control. *Proceedings of the VLDB Endowment*, 2017.

[89] Sayed Hadi Hashemi, Faraz Faghri, Paul Rausch, and Roy H Campbell. World of empowered IoT users. In *First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2016.

[90] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A survey on IoT security: application areas, security threats, and solution architectures. *Access Journal*, 2019.

[91] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. Rethinking access control and authentication for the home Internet of Things (IoT). In *27th USENIX Security Symposium*, 2018.

[92] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. Rethinking access control and authentication for the home internet of things (IoT). In *USENIX Security Symposium*, 2018.

[93] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. Rethinking access control and authentication for the home internet of things (IoT). In *27th {USENIX} Security Symposium*, 2018.

[94] Weijia He, Valerie Zhao, Olivia Morkved, Sabeeka Siddiqui, Earlence Fernandes, Josiah Hester, and Blase Ur. SoK: Context sensing for access control in the adversarial home IoT.

In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 37–53. IEEE, 2021.

[95] Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin. Smart nest thermostat: A smart spy in your home. *Black Hat USA*, 2014.

[96] José L Hernández-Ramos, Antonio J Jara, Leandro Marín, and Antonio F Skarmeta Gómez. DCapBAC: embedding authorization logic into smart things through ECC optimizations. *International Journal of Computer Mathematics*, 93(2):345–366, 2016.

[97] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. Smart locks: Lessons for securing commodity internet of things devices. In *Proceedings of the 11th ACM on Asia conference on computer and communications security (ASIACCS)*, 2016.

[98] Vincent Hu. Blockchain for access control systems. Technical report, National Institute of Standards and Technology (NIST), 2021.

[99] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (ABAC) definition and considerations (draft). *NIST special publication*, 800(162):1–54, 2013.

[100] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo. Access control for emerging distributed systems. In *IEEE Computer*, 2018.

[101] IEEE. Ieee standard for an architectural framework for the internet of things (IoT).

[102] RFC 6749 Internet Engineering Task Force (IETF). The OAuth 2.0 authorization framework, 2012. Available at `https://www.rfc-editor.org/rfc/pdfrfc/rfc6749.txt.pdf`.

[103] Marian K Iskander, Dave W Wilkinson, Adam J Lee, and Panos K Chrysanthis. Enforcing policy and data consistency of cloud transactions. In *Proceedings of 31st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 253–262. IEEE, 2011.

[104] MD Azharul Islam and Sanjay Madria. A permissioned blockchain based access control system for IoT. In *Blockchain*. IEEE, 2019.

[105] Abhiditya Jha, Jess Kropczynski, Heather Richter Lipford, and Pamela J Wisniewski. An exploration on sharing smart home devices beyond the home. In *IUI Workshops*, 2019.

[106] Shravya Kanchi and Kamalakar Karlapalem. A multi perspective access control in a smart home. In *Conference on Data and Application Security and Privacy (CODASPY)*, 2021.

[107] Ji Eun Kim, Tassilo Barth, George Boulos, John Yackovich, Christian Beckel, and Daniel Mosse. Seamless integration of heterogeneous devices and access control in smart homes and its evaluation. *Intelligent Buildings International*, 2017.

[108] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. Challenges in access right assignment for secure home networks. In *Workshop on Hot Topics in Security (HotSec 10)*. USENIX, 2010.

[109] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. Challenges in access right assignment for secure home networks. In *HotSec*, 2010.

[110] Yki Kortesniemi and Mikko Sarela. Survey of certificate usage in distributed access control. In *Elsevier Journal of Computers and Security*, volume 44, pages 16–32. Elsevier, 2014.

[111] Ram Krishnan, Jianwei Niu, Ravi Sandhu, and William H Winsborough. Stale-safe security properties for group-based secure information sharing. In *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, pages 53–62. ACM, 2008.

[112] Ram Krishnan and Ravi Sandhu. Authorization policy specification and enforcement for group-centric secure information sharing. In *International Conference on Information Systems Security*, pages 102–115. Springer, 2011.

[113] Nir Kshetri. Can blockchain strengthen the internet of things? *IT professional*, 2017.

[114] Adam J Lee, Kazuhiro Minami, and Nikita Borisov. Confidentiality-preserving distributed proofs of conjunctive queries. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 287–297. ACM, 2009.

[115] Adam J Lee, Kazuhiro Minami, and Marianne Winslett. Lightweight consistency enforcement schemes for distributed proofs with hidden subtrees. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 101–110. ACM, 2007.

[116] Adam J Lee and Marianne Winslett. Safety and consistency in policy-based authorization systems. In *Proceedings of the 13th ACM conference on Computer and communications security*, 2006.

[117] Adam J Lee and Ting Yu. Towards quantitative analysis of proofs of authorization: applications, framework, and techniques. In *23rd IEEE Computer Security Foundations Symposium (CSF)*, pages 139–153. IEEE, 2010.

[118] Sanghak Lee, Jiwon Choi, Jihun Kim, Beumjin Cho, Sangho Lee, Hanjun Kim, and Jong Kim. FACT: Functionality-centric access control system for iot programming frameworks. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies (SACMAT)*, 2017.

[119] Gerhard Leitner, Anton Fercher, Alexander Felfernig, Klaus Isak, S Polat Erdeniz, Arda Akcay, and Michael Jeran. Recommending and configuring smart home installations. In *International Workshop on Configuration (ConfWS)*, 2016.

[120] Ninghui Li and Ziqing Mao. Administration in role-based access control. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 127–138, 2007.

[121] Olof Liberg. The Cellular Internet of Things.

[122] Zhen Ling, Junzhou Luo, Yiling Xu, Chao Gao, Kui Wu, and Xinwen Fu. Security vulnerabilities of internet of things: A case study of the smart plug system. *Internet of Things Journal*, 2017.

[123] Han Liu, Dezhi Han, and Dun Li. Fabric-IoT: A blockchain-based access control system in IoT. *Access*, 2020.

[124] Emil C Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *Transactions on software engineering*, 25(6):852–869, 1999.

[125] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In *IFIP international conference on distributed applications and interoperable systems*. Springer, 2017.

[126] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. A blockchain based approach for the definition of auditable access control systems. *Computers & Security*, 2019.

[127] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. Unlocking the Potential of the Internet of Things. *McKinsey Global Institute*, 2015.

[128] Antonio La Marra, Fabio Martinelli, Paolo Mori, Athanasios Rizos, and Andrea Saracino. Improving MQTT by inclusion of usage control. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage(SpaCCS)*. Springer, 2017.

[129] Juri Mattila. The blockchain phenomenon–the disruptive potential of distributed consensus architectures. Technical report, ETLA working papers, 2016.

[130] Katerina Megas, Barbara Cuthill, and Sarbari Gupta. Establishing confidence in iot device security: How do we get there?(draft). Technical report, National Institute of Standards and Technology (NIST), 2021.

[131] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. Iot: Internet of threats? a survey of practical security vulnerabilities in real IoT devices. *Internet of Things Journal*, 2019.

[132] Dominik Meyer, Jan Haase, Marcel Eckert, and Bernd Klauer. A threat-model for building and home automation. In *14th international conference on INDustrial INformatics (INDIN)*. IEEE, 2016.

[133] Ziarmal Nazar Mohammad, Fadi Farha, Adnan OM Abuassba, Shunkun Yang, and Fang Zhou. Access control and authorization in smart homes: A survey. *Tsinghua Science and Technology*, 2021.

[134] Satoshi Nakamoto. Bitcoin whitepaper. *Available at* `https://bitcoin.org/bitcoin.pdf`, 2008.

[135] Yuta Nakamura, Yuanyu Zhang, Masahiro Sasabe, and Shoji Kasahara. Capability-based access control for the internet of things: an Ethereum blockchain-based scheme. In *Global Communications Conference (GLOBECOM)*. IEEE, 2019.

[136] Antonio L Maia Neto, Artur LF Souza, Italo Cunha, Michele Nogueira, Ivan Oliveira Nunes, Leonardo Cotta, Nicolas Gentille, Antonio AF Loureiro, Diego F Aranha, Harsh Kupwade Patil, et al. AOT: Authentication and access control for the entire IoT device life-cycle. In *Conference on Embedded Network Sensor Systems CD-ROM (SenSys)*. ACM, 2016.

[137] RFC 5280 Network Working Group. Internet X.509 public key infrastructure certificate (PKI) and certificate revocation list (CRL) profile, 2008. Available at https://www.rfc-editor.org/rfc/pdfrfc/rfc5280.txt.pdf.

[138] Qun Ni, Elisa Bertino, and Jorge Lobo. Risk-based access control systems built on fuzzy inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 250–260. ACM, 2010.

[139] Sukhvir Notra, Muhammad Siddiqi, Hassan Habibi Gharakheili, Vijay Sivaraman, and Roksana Boreli. An experimental study of security and privacy risks with emerging household appliances. In *CNS*. IEEE, 2014.

[140] Mahda Noura, Mohammed Atiquzzaman, and Martin Gaedke. Interoperability in internet of things: Taxonomies and open challenges. *Mobile networks and applications*, 2019.

[141] Oscar Novo. Blockchain meets IoT: An architecture for scalable access management in IoT. *Internet of Things Journal*, 2018.

[142] OASIS. Security Assertion Markup Language (SAML) V2.0 Technical Overview, 2008. Available at http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.pdf.

[143] Se-Ra Oh, Young-Gab Kim, and Sanghyun Cho. An interoperable access control framework for diverse IoT platforms based on OAuth and role. *Sensors*, 2019.

[144] B Cli ord Neuman. Scale in distributed systems. *ISI/USC*, page 68, 1994.

[145] Aafaf Ouaddah, Anas Abou El Kalam, and Abdellah Ait Ouahman. Harnessing the power of blockchain technology to solve IoT security & privacy issues. In *ICC*, 2017.

[146] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the Internet of Things. *Security and communication networks*, 2016.

[147] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In *Europe and MENA cooperation advances in information and communication technologies*. Springer, 2017.

[148] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, and Abdellah Ait Ouahman. Access control in the internet of things: Big challenges and new opportunities. *Computer Networks*, 2017.

[149] Federica Paci, Massimo Mecella, Mourad Ouzzani, and Elisa Bertino. ACConv–an access control model for conversational web services. *ACM Transactions on the Web (TWEB)*, 5(3):13, 2011.

[150] Jaehong Park and Ravi Sandhu. The UCON_ABC usage control model. *Transactions on information and system security (TISSEC)*, 2004.

[151] Sean Peisert, Ed Talbot, and Matt Bishop. Turtles all the way down: a clean-slate, ground-up, first-principles approach to secure systems. In *Proceedings of the 2012 New Security Paradigms Workshop*, pages 15–26. ACM, 2012.

[152] Víctor Peláez, Roberto González, Luis Ángel San Martín, Antonio Campos, and Vanesa Lobato. Multilevel and hybrid architecture for device abstraction and context information management in smart home environments. In *International Joint Conference on Ambient Intelligence*. Springer, 2010.

[153] Charith Perera, Prem Prakash Jayaraman, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context-aware dynamic discovery and configuration of âthingsâ in smart environments. In *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014.

[154] Matthieu Perrin. *Distributed systems: concurrency and consistency*. Elsevier, 2017.

[155] Otto Julio Ahlert Pinno, Andre Ricardo Abed Gregio, and Luis CE De Bona. Controlchain: Blockchain as a central enabler for access control authorizations in the IoT. In *GLOBECOM Global Communications Conference*. IEEE, 2017.

[156] Geong Sen Poh, Prosanta Gope, and Jianting Ning. Privhome: Privacy-preserving authenticated communication in smart home environment. *IEEE Transactions on Dependable and Secure Computing*, 18(3):1095–1107, 2019.

[157] Jing Qiu, Zhihong Tian, Chunlai Du, Qi Zuo, Shen Su, and Binxing Fang. A survey on access control in the age of internet of things. *Internet of Things Journal*, 2020.

[158] Jing Qiu, Zhihong Tian, Chunlai Du, Qi Zuo, Shen Su, and Binxing Fang. A survey on access control in the age of internet of things. *Internet of Things Journal*, 2020.

[159] Mohsin Ur Rahman. Scalable role-based access control using the EOS blockchain. *arXiv preprint arXiv:2007.02163*, 2020.

[160] Sowmya Ravidas, Alexios Lekidis, Federica Paci, and Nicola Zannone. Access control in Internet-of-Things: A survey. *Journal of Network and Computer Applications*, 2019.

[161] Indrakshi Ray, Ramadan Abdunabi, and Rejina Basnet. Access control for internet of things applications. In *Workshop on Cyber-Physical System Security*. ACM, 2019.

[162] CORDIS EU Research Results. Big iot-bridging the interoperability gap of the Internet of Things.

[163] CORDIS EU Research Results. Building an IoT OPen innovation Ecosystem for connected smart objects.

[164] CORDIS EU Research Results. symbIoTe-Symbiosis of smart objects across IoT environments.

[165] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. challenges and opportunities. *Future generation computer systems*, 2018.

[166] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin OâFlynn. Iot goes nuclear: Creating a zigbee chain reaction. In *Symposium on Security and Privacy (S&P)*. IEEE, 2017.

[167] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin OâFlynn. Iot goes nuclear: Creating a ZigBee chain reaction. In *Symposium on Security and Privacy (SP)*. IEEE, 2017.

[168] Sara Rouhani and Ralph Deters. Blockchain based access control systems: State of the art and challenges. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019.

[169] Ravi Sandhu. The PEI framework for application-centric security. In *Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2009.

[170] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *Transactions on Information and System Security (TISSEC)*, 1999.

[171] Ravi Sandhu and Jaehong Park. Usage control: A vision for next generation access control. In *MMM-ACNS*. Springer, 2003.

[172] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 1996.

[173] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.

[174] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *Communications Magazine*, 1994.

[175] Stefan Schmid, Arne Bröring, Denis Kramer, Sebastian Käbisch, Achille Zappa, Martin Lorenz, Yong Wang, Andreas Rausch, and Luca Gioppo. An architecture for interoperable iot ecosystems. In *Interoperability and Open-Source Solutions*. Springer, 2016.

[176] Savio Sciancalepore et al. On the design of a decentralized and multiauthority access control scheme in federated and cloud-assisted cyber-physical systems. In *IEEE Internet of Things Journal*, 2018.

[177] Savio Sciancalepore, Giuseppe Piro, Daniele Caldarola, Gennaro Boggia, and Giuseppe Bianchi. OAuth-IoT: An access control framework for the Internet of Things based on open standards. In *Symposium on Computers and Communications (ISCC)*. IEEE, 2017.

[178] Savio Sciancalepore, Giuseppe Piro, Daniele Caldarola, Gennaro Boggia, and Giuseppe Bianchi. On the design of a decentralized and multiauthority access control scheme in federated and cloud-assisted cyber-physical systems. *Internet of Things Journal*, 2018.

[179] Byoungjin Seok, Jose Costa Sapalo Sicato, Tcydenova Erzhena, Canshou Xuan, Yi Pan, and Jong Hyuk Park. Secure d2d communication for 5g iot network based on lightweight cryptography. *Applied Sciences*, 2020.

[180] Martin Serrano, Hoan Nguyen Mau Quoc, Danh Le Phuoc, Manfred Hauswirth, John Soldatos, Nikos Kefalakis, Prem Prakash Jayaraman, and Arkady Zaslavsky. Defining the stack for service delivery models and interoperability in the Internet of Things: A practical case with OpenIoT-VDK. *IEEE Journal on Selected Areas in Communications*, 33(4):676–689, 2015.

[181] Kinza Shafique, Bilal A Khawaja, Farah Sabir, Sameer Qazi, and Muhammad Mustaqim. Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios. *Access*, 2020.

[182] Kinza Shafique, Bilal A Khawaja, Farah Sabir, Sameer Qazi, and Muhammad Mustaqim. Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios. *Access Journal*, 2020.

[183] Mehrnoosh Shakarami and Ravi Sandhu. Refresh instead of revoke enhances safety and availability: A formal analysis. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 301–313. Springer, 2019.

[184] Mehrnoosh Shakarami and Ravi Sandhu. Safety and consistency of mutable attributes using quotas: A formal analysis. In *Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2019.

[185] Mehrnoosh Shakarami and Ravi Sandhu. Safety and consistency of subject attributes for attribute-based pre-authorization systems. In *National Cyber Summit*, pages 248–263. Springer, 2019.

[186] Mehrnoosh Shakarami and Ravi Sandhu. Role-based administration of role-based smart home IoT. In *Workshop on Secure and Trustworthy Cyber-Physical Systems*. ACM, 2021.

[187] Amit Kumar Sikder, Leonardo Babun, Z Berkay Celik, Abbas Acar, Hidayet Aksu, Patrick McDaniel, Engin Kirda, and A Selcuk Uluagac. Kratos: multi-user multi-device-aware access control system for the smart home. In *Security and Privacy in Wireless and Mobile Networks*. ACM, 2020.

[188] John Soldatos, Nikos Kefalakis, Manfred Hauswirth, Martin Serrano, Jean-Paul Calbimonte, Mehdi Riahi, Karl Aberer, Prem Prakash Jayaraman, Arkady Zaslavsky, Ivana Podnar Žarko, et al. OpenIoT: Open source internet-of-things in the cloud. In *Interoperability and open-source solutions for the internet of things*, pages 13–25. Springer, 2015.

[189] Sergios Soursos, Ivana Podnar Žarko, Patrick Zwickl, Ivan Gojmerac, Giuseppe Bianchi, and Gino Carrozzo. Towards the cross-domain interoperability of IoT platforms. In *2016 European conference on networks and communications (EuCNC)*. IEEE, 2016.

[190] Anna C Squicciarini, Alberto Trombetta, Elisa Bertino, and Stefano Braghin. Identity-based long running negotiations. In *Proceedings of the 4th ACM workshop on Digital identity management*, pages 97–106. ACM, 2008.

[191] Mark Stanislav and Tod Beardsley. Hacking IoT: A case study on baby monitor exposures and vulnerabilities. *Rapid7 Report*, 2015.

[192] Martin van Steen and Andrew S. Tanenbaum. *Distributed Systems*. Martin van Steen, 2017.

[193] Madiha Tabassum, Jess Kropczynski, Pamela Wisniewski, and Heather Richter Lipford. Smart home beyond the home: A case for community-based access control. In *CHI Conference on Human Factors in Computing Systems*, 2020.

[194] Madiha Tabassum, Jess Kropczynski, Pamela Wisniewski, and Heather Richter Lipford. Smart home beyond the home: A case for community-based access control. In *CHI Conference on Human Factors in Computing Systems*. ACM, 2020.

[195] Liang Tan, Na Shi, Keping Yu, Moayad Aloqaily, and Yaser Jararweh. A blockchain-empowered access control framework for smart devices in green Internet of Things. *Transactions on Internet Technology (TOIT)*, 2021.

[196] Pietro Tedeschi, Giuseppe Piro, Jose Antonio Sanchez Murillo, Nemanja Ignjatov, Michał Pilc, Kaspar Lebloch, and Gennaro Boggia. Blockchain as a service: Securing bartering functionalities in the H2020 symbIoTe framework. *Internet Technology Letters*, 2019.

[197] The Global Community that Develops Standards for IoT. onem2m:standards for m2m and the internet of things.

[198] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. Smartauth: User-centered authorization for the internet of things. In *26th {USENIX} Security Symposium*, 2017.

[199] Petar Tsankov, Srdjan Marinovic, Mohammad Torabi Dashti, and David Basin. Fail-secure access control. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1157–1168. ACM, 2014.

[200] Hannes Tschofenig, Jari Arkko, Dave Thaler, and D McPherson. Architectural considerations in smart object networking. *RFC 7452*, 2015.

[201] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *Computer*, 36(10):38–44, 2003.

[202] Blase Ur, Jaeyeon Jung, and Stuart Schechter. The current state of access control for smart devices in homes. In *Workshop on Home Usable Privacy and Security (HUPS)*. HUPS, 2013.

[203] Qihua Wang and Hongxia Jin. Quantified risk-adaptive access control for patient privacy protection in health information systems. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 406–410. ACM, 2011.

[204] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[205] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018.

[206] Min Xu, Duminda Wijesekera, Xinwen Zhang, and Deshan Cooray. Towards session-aware RBAC administration and enforcement with XACML. In *International Symposium on Policies for Distributed Systems and Networks*. IEEE, 2009.

[207] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen. BlendCAC: A blockchain-enabled decentralized capability-based access control for IoTs. In *Internet of Things (iThings)*. IEEE, 2018.

[208] Kan Yang et al. DAC-MACS: Effective data access control for multiauthority cloud storage systems. In *IEEE Information Forensics and Security*, 2013.

[209] Kan Yang and Xiaohua Jia. Attributed-based access control for multi-authority systems in cloud storage. In *IEEE ICDCS*, 2012.

[210] Kan Yang and Xiaohua Jia. Expressive, efficient, and revocable data access control for multi-authority cloud storage. In *IEEE Parallel and Distributed Systems*, 2014.

[211] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao. A survey on security and privacy issues in internet-of-things. *Internet of Things Journal*, 2017.

[212] Ehtesham Zahoor et al. Authorization policies specification and consistency management within multi-cloud environments. In *NordSec*. Springer, 2018.

[213] Eric Zeng, Shrirang Mare, and Franziska Roesner. End user security and privacy concerns with smart homes. In *thirteenth symposium on usable privacy and security ({SOUPS})*, 2017.

[214] Eric Zeng and Franziska Roesner. Understanding and improving security and privacy in multi-user smart homes: a design exploration and in-home user study. In *28th {USENIX} Security Symposium*, 2019.

[215] Eric Zeng and Franziska Roesner. Understanding and improving security and privacy in multi-user smart homes: a design exploration and in-home user study. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019.

[216] Guoping Zhang and Jiazheng Tian. An extended role based access control model for the Internet of Things. In *International Conference on Information, Networking and Automation (ICINA)*. IEEE, 2010.

[217] Yuanyu Zhang, Shoji Kasahara, et al. Smart contract-based access control for the internet of things. *Internet of Things Journal*, 2018.

[218] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. Discovering and understanding the security hazards in the interactions between {IoT} devices, mobile apps, and clouds on smart home platforms. In *28th USENIX Security Symposium*, 2019.

VITA

Mehrnoosh Shakarami was born and grew up in Khorram Abad, Lorestan, Iran. She earned her Bachelor's Degree in Computer Software Engineering in August 2005 from Yazd University, Yazd, Iran. She graduated with a Master's degree in Computer Software Engineering from Sharif University of Technology, Tehran, Iran in Feb 2009. She worked as faculty member, instructor of core Computer Science courses and group manager in the Computer Engineering group of Lorestan University for six years. She has been admitted to the PhD Program in Computer Science in August 2015 at State University of New York at Binghamton (SUNY-Binghamton). In Spring 2018, Mehrnoosh transferred her studies to University of Texas at San Antonio (UTSA) where she joined Institute for Cyber Security (ICS) as a PhD student in the Department of Computer Science. She started her research under supervision of Professor Ravi Sandhu, focused on operation and administration of access control models in IoT environments. She also received a Master's degree in Computer Science from UTSA with specialization in Computer and Information Security.