ACCESS CONTROL MODELS FOR CLOUD-ENABLED INTERNET OF THINGS

by

ASMA HASSAN ALSHEHRI, 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:
Dr. Ravi Sandhu, Ph.D., Chair
Dr. Gregory B. White, Ph.D.
Dr. Matthew Gibson, Ph.D.
Dr. Palden lama, Ph.D.
Dr. Ram Krishnan, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
Mar 2018

DEDICATION

I dedicate this work to my family and friends. First, I dedicate it to the greatest parents, Hassan and Zeina, who brought me to the life, raised me, and believed in my ability to accomplish my Ph.D. degree. Also, I dedicate this work to the kindest and best husband, Ahmed Alshehri, who supported me during my study and gave me enormous time to concentrate on my dissertation. I dedicate it also to my siblings Muhammad, Fatimah, Reem, Riyadh, and Abdul Majeed who supported and encouraged me to successfully complete my study. Finally, I dedicate it to my lovely prince Hassan and my little princess Zeina who are my source of strength.

ACKNOWLEDGEMENTS

I would like to give special thanks to people who assisted me to achieve this dissertation. I would like to express the deepest appreciation to my committee chair Dr. Sandhu, who continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this dissertation would not have been possible. Dr. Sandhu helped me with his inspiring ideas, critical comments, and constant encouragement that helped me overcome difficulties during my researches. Dr. Sandhu was not only a professor who care about my researches, but like a father who care about my family life. I am thankful to him for his guidance in every professional and personal aspects throughout my doctoral studies and beyond.

I would like to especially acknowledge the support and help from Mr. Farhan Patwa, who is the associate director in the Institute for Cyber Security (ICS) of UTSA. I would like to thank Mr. Farhan Patwa for his guidance and expertise regarding implementing my models in Amazon Web Services (AWS) for the Internet of Things (IoT). Also, He also helped me in studying and learning OpenStack platform.

I also would like to thank my committee members Dr. Gregory B. White, Dr. Palden lama, Dr. Matthew Gibson, and Dr. Ram Krishnan for their valuable knowledge, comments, time, and insight in organizing this dissertation.

Also, I would like to thank Mr. James Benson for his collaboration in implementing my models in Amazon Web Services (AWS) for the Internet of Things (IoT).

I would like to acknowledge our faculties from the Computer Science department, Suzanne Tanaka from the ICS, Susan Allen and other staffs from CS department for their support throughout my doctoral studies.

I am grateful to all of those whom I have had the pleasure to work with during this and other related work. I want to thank all my fellows (some of them already achieved their Ph.D degrees) at UTSA, especially ICS fellows. They provided me a great help during the process of searching

for ideas and implementing them.

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 2018

ACCESS CONTROL MODELS FOR CLOUD-ENABLED INTERNET OF THINGS

Asma Hassan Alshehri, Ph.D.
The University of Texas at San Antonio, 2018

Supervising Professor: Dr. Ravi Sandhu, Ph.D.

The concept and deployment of Internet of Things (IoT) has continued to develop momentum over recent years. The rapid development of IoT in recent years has triggered a wave of potentially unreasonable expectations. Many industries have started big projects with key technologies that incorporate the basic architecture of IoT, which has not been determined yet. Several different layered architectures for IoT have been proposed. In general, the proposed IoT architectures comprise three main components: an object layer, one or more middle layers, and an application layer. The main difference in detail between them is in the middle layers. Some architectures include a cloud services layer for managing IoT things. Some suggest the use of virtual objects as digital counterparts for physical IoT objects. Sometimes both cloud services and virtual objects are included.

In this dissertation, we initially propose an IoT architecture that can be used to develop an authoritative family of access control models for a cloud-enabled Internet of Things. Our proposed access-control oriented (ACO) architecture for IoT comprises four layers: an object layer, a virtual object layer, a cloud services layer, and an application layer. This 4-layer architecture serves as a framework to build access control models for a cloud-enabled IoT. Within this architecture, we present illustrative examples that highlight some IoT access control issues leading to a discussion of needed access control research. We identify the need for communication control within each layer and across adjacent layers (particularly in the lower layers), coupled with the need for data access control (particularly in the cloud services and application layers)

The ACO architecture is proposed for the cloud-enabled IoT, with virtual objects (VOs) and cloud services in the middle layers. A central aspect of ACO is to control communication among VOs. To this end, we develop operational and administrative access control models, assuming

topic-based publish-subscribe interaction among VOs. Operational models are developed using (i) access control lists for topics and capabilities for virtual objects, and (ii) attribute-based access control, and it is argued that role-based access control is not suitable for this purpose. Administrative models for these two operational models are developed using (i) access control lists, (ii) role-based access control, and (iii) attribute-based access control. A use case of sensing speeding cars illustrates the details of these access control models for VO communication, and their differences. An assessment of these models with respect to security and privacy preserving objectives of IoT is also provided.

Finally, we study AWS IoT as a major commercial cloud-IoT platform and investigate its suitability for implementing the afore-mentioned academic models of ACO and VO communication control. While AWS IoT has a notion of digital shadows closely analogous to VOs, it lacks explicit capability for VO communication and thereby for VO communication control. Thus, there is a significant mismatch between AWS IoT and these academic models. Our principal contribution in this regard is to reconcile this mismatch by showing how to use the mechanisms of AWS IoT to effectively implement VO communication models. To this end, we develop an access control model for virtual objects (shadows) communication in AWS IoT called AWS-IoT-ACMVO. We develop a proof-of-concept implementation of the speeding cars use case in AWS IoT under guidance of this model, and provide selected performance measurements. We conclude with a discussion of possible alternate implementations of this use case in AWS IoT.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: INTRODUCTION

In recent few decades, the concept of Internet of Things (IoT) has appeared in our lives and has lately attracted the attention of governments, companies, and academia. The IoT is an extension of network technology, where the main core of communication is the Internet. The promising Internet of Things paradigm integrates many widely dispersed, mobile, abundant, heterogeneous objects, such as sensors and actuators that collect data from an environment. It also incorporates various technologies, such as Internet, network, communication protocols, and cloud and mobile computing.

The rapid growth of IoT last years has triggered a wave of potentially unrealistic expectations. For instance, many businesses have initiated big projects with key technologies that integrate the basic architecture of IoT, which has not been determined yet. Thus, there is a fundamental need to improve and determine one or more standard architectures for the upcoming IoT.

Most IoT architectures in industry combine the IoT with major cloud services that already exist. Also, various academic IoT architectures suggested integrating the cloud within IoT architecture to benefit from the cloud capability such as offering virtually unlimited computational capabilities, low-cost, and on-demand storage capacity. This integration is referred to cloud-based IoT, cloud-assisted IoT, or cloud-enabled IoT, which we adopt in this dissertation.

The cloud-enabled IoT has obtained much popularity in industry and academia. Most of academia researches focus on either certain applications and technologies of the IoT or on IoT surveys. However, one of the main issues of introducing the IoT technology into the real word is security and privacy. Because the IoT architecture encompasses pervasive connected heterogeneous objects, which interact with each other, with applications, and with other entities, security is necessary for its wide adoption and continued success. In this dissertation, we focus on access control models for the cloud-enabled IoT, mainly authorization for virtual objects to communicate with each other.

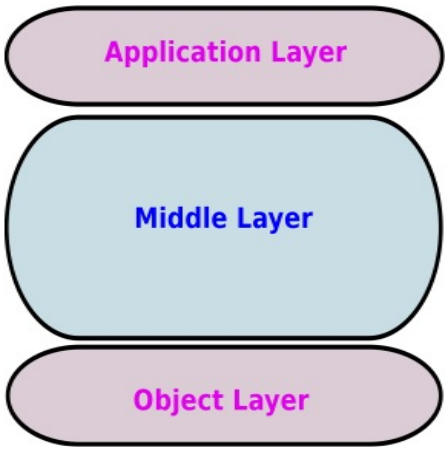

Figure 1.1: General IoT Architecture from Literature

1.1 Motivation

In last few years, the academic and industrial communities have witnessed the rapid growth of the concept of IoT. The many widely connected entities through Internet such as objects and applications make issues within the IoT platform raise more complicated and different issues compared with traditional distributed systems. Thus, there is a need to adjust current solutions or propose new ones in order to handle IoT issues.

1.1.1 The IoT Architecture

The rapid and fast growth of IoT needs to be built upon specific classification and architecture. However, many businesses have introduced their IoT projects upon their view of IoT. That led to have various IoT projects with different classifications. In other words, one IoT issue that may not happen with one business may appear with other businesses. Therefore, there is an essential need to determine one or more standard architectures for the upcoming IoT.

There have been various proposals for IoT architecture, although none so far is regarded as the definitive IoT architecture. To our knowledge, all the proposed IoT architectures that we have seen are divided into three main layers: an object layer, one or more middle layers, and an application layer as shown in Figure 1.1. The main difference in detail between them is in the middle layers. Some architectures abstracted the middle layers to only one layer [76], while others have two or

more middle layers [1, 8, 58, 74].

Various research papers [1, 8, 58, 61, 74] suggested integrating the cloud with the IoT as a central entity between object and application layer. The IoT can benefit from the powerful capabilities and resources of the cloud to support and compensate its technological constraints. The IoT encompasses pervasive and heterogeneous objects that produce big non-structured or semi-structured data. IoT objects have limited computational power and low storage capacity. Offering virtually unlimited computational capabilities, low-cost, on-demand storage capacity, and ubiquitous resources usable from everywhere. It has been suggested using the cloud as the most convenient and cost-effective solution to deal with IoT technological constraints [13, 14, 57, 60].

The IoT concept is developed within various technologies and many dimensions to support distributed smart objects to collaborate and provide real-world intelligence. This complex environment is faced with list of major IoT issues that need to be handled regardless of building a formal IoT architecture. Scalability, heterogeneity, constrained resource, mobility, identification, and security and privacy are examples of major issues that the IoT ecosystem may face.

Incorporating virtual object layer as an essential part of IoT architecture has been discussed in depth in [52] with respect to general IoT issues. Atzoori et al. [8] argue for such a layer to unite access to the heterogeneous devices in the object layer. Evangelos et al. [24] explain the benefit of exposing the capabilities and services of objects to the upper layers through such abstractions. A comparable definition as [24] is given in [68] with the name of 'virtual object layer'. A virtual object is also described in [73] as comprised of both current and historical information about a specific physical object. A virtual object is called a device shadow in Amazon Web Service (AWS) IoT, which persists the final and desired future status of each device, even when the device is offline. Thus, we can see that integrating the cloud layer and the virtual object layer with the IoT architecture supports and compensates IoT technological constraints.

1.1.2 The IoT Security Issues

One of the main components of the IoT is physical objects that impact on human beings on the levels of personal daily lives to sophisticated financial transactions that are vital to global business. Different entities are connected to each other in the IoT such as objects, networks, cloud, or applications need to be secured. There are security issues within IoT that need to be handled. Data authentication, for example, where object information must be authenticated is important issue to address. Access control is also another issue where data and entities can only be accessed by other authorized entities. Such issues should be addressed and handled.

In fact, there have been various work proposed access control solutions for IoT based on certain scenarios and use cases [15, 30, 47]. However, most proposed scenarios are not following specific and formal architecture to be built upon. Thus, addressing security issues, such as access control, of the IoT is not simple to be determined without following a suitable and clear IoT architecture.

1.2 Problem Statement

A robust IoT architecture is necessary to be build IoT upon. Integrating the cloud as a central layer within the IoT architecture has been suggested to offset IoT technological constraints. In other words, the IoT can benefit from the powerful capabilities and resources of the cloud. Also, incorporating virtual object layer within IoT architecture has been suggested to mitigate IoT major issues, such as scalability, heterogeneity, constrained resource, security and privacy, and identification. Thus, integrating both of virtual object and cloud layer in the middle layer of the IoT architecture helps in supporting and compensating IoT technological constraints.

The growth of collected data that flows through other components of IoT is exposed to exploitation. One way of controlling access to the collected data from physical objects is to control the communication among IoT physical objects, which can communicate in a distributed manner. Because of pervasive heterogeneous physical objects, studying virtual objects communication is more practical where objects can communicate in a homogeneous manner. This kind of communication needs to be controlled, so the security of virtual objects and their collected data can be

preserved. The access control models that have been proposed can be used and adjusted appropriately to accommodate IoT needs.

1.3 Thesis Statement

An initial set of access control models in cloud-enabled Internet of Things can be developed within a robust framework, which can support further maturation and elaboration of this initial set. Integrating the virtual object and the cloud layer within the IoT architecture is valuable to offset IoT issues. Virtual objects communication offers easier communication of the heterogeneous objects, so developing and controling virtual objects communication is useful for IoT success. Virtual object communication can be controlled by adjusting the traditional access control models, which are studied and reconciled within the AWS IoT.

1.4 Summary of Contribution

The major contributions of our dissertation are summarized in Figure1.2 and described as follows:

- Propose IoT architecture that is built upon previous proposed IoT architectures and suggestions of using cloud and virtual object concept within IoT. The proposed IoT architecture is called access control oriented (ACO) architecture for the cloud-enabled IoT, which incorprates various IoT security and access control issues.

- Investigating the communication among virtual objects and introduce ways to control this communication based on ACO architecture for the cloud-enabled IoT. Existing traditional access control models are adapted and utilized to develop operational models and administrative models for the proposed operational models, referred to as ACO-IoT-ACMsVO. A use case of sensing speeding cars illustrates the details of these access control models for VO communication.

  - Operational models are developed using (i) access control lists for topics and capabilities for virtual objects, and (ii) attribute-based access control. It is argued that role-

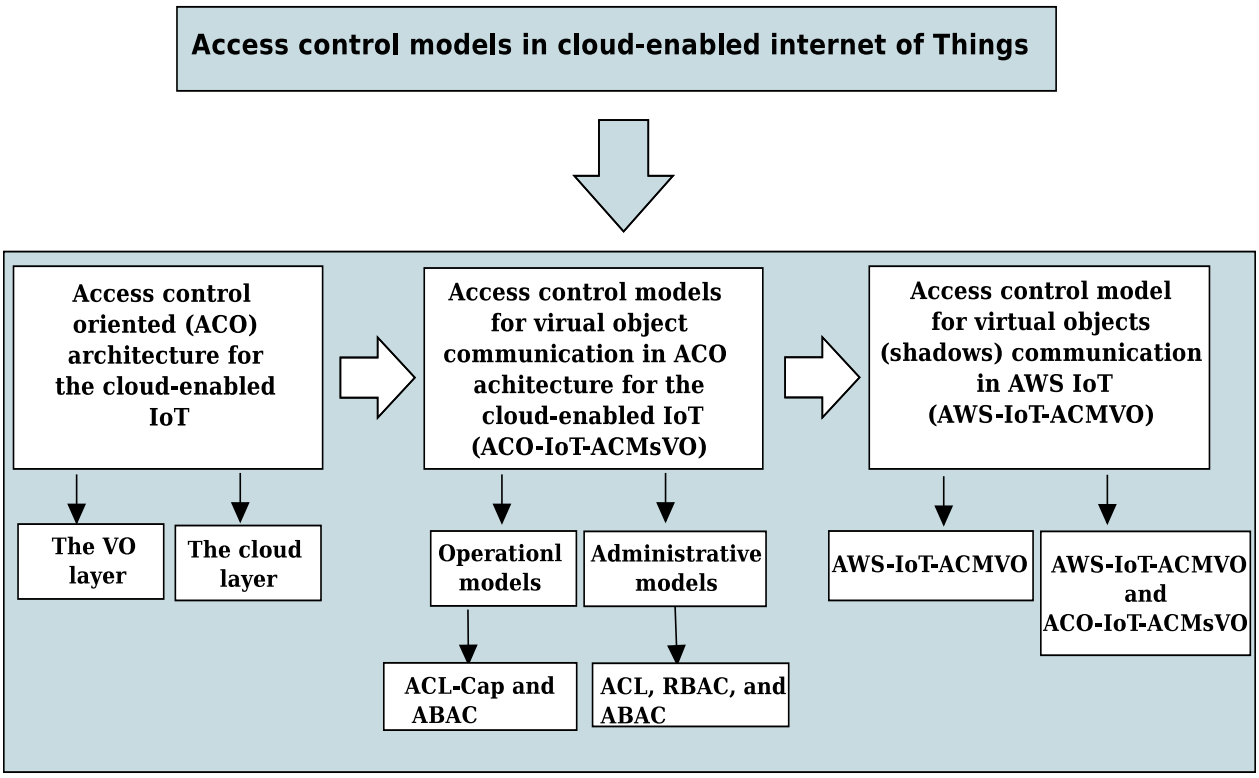

Figure 1.2: Outline of Contributions

based access control is not suitable for this purpose.

  - Administrative models for the two operational models are developed using (i) access control lists, (ii) role-based access control, and (iii) attribute-based access control.

- Investigating AWS IoT as a major commercial cloud-IoT platform and demonstrate its suitability for implementing our operational models of VO communication. This lead to develop an access control model for virtual objects (shadows) communication in AWS IoT called AWS-IoT-ACMVO.

1.5 Scope and Assumptions

We take a first step toward our eventual goal of evolving an authoritative family of access control models for a cloud-enabled Internet of Things. Our proposed access control oriented (ACO) architecture for the cloud-enabled IoT comprises four layers: an object layer, a virtual object layer,

a cloud services layer, and an application layer. Each of these layers encapsulate different entities, associated data, and access control requirements. The ACO architecture will be our reference to build access control models for a cloud-enabled IoT. The assumption and scope of the initial proposed access control models for the IoT is as follow.

- There are different kinds of communications within our ACO architecture for the cloud-enabled IoT. In this dissertation, we propose access control models that focus on controlling the communication among virtual objects in the virtual object layer. Other kinds of communications within ACO architecture can be investigated in future works.

- The physical objects within our ACO architecture for the cloud-enabled IoT can communicate with each other or with the upper layer. In our access control models for virtual object communication, we assumed that physical objects can only communicate with each other through their virtual objects in the virtual object layer.

- The cloud layer within our ACO architecture can have one or multiple clouds. In our access control models for virtual object communication, we assumed that there is only one central cloud. Thus, one centralized cloud is assumed in the cloud layer of the ACO architecture, and there is no consideration for multi-clouds collaboration.

1.6 Organization of the Dissertation

Chapter 2 reviews academic published IoT architectures, IoT issues, access control models, and recent access control models for IoT. Chapter 3 introduces ACO architecture for the cloud-enabled IoT that used as a foundation to study IoT security issues, develop operational access control models, and administrative access control models. In Chapter 4, a access control models for virtual object communication in cloud-enabled IoT is proposed and explained in details within a use case, referred to as ACO-IoT-ACMsVO. Chapter 5 reconciles ACO-IoT-ACMsVO within AWS IoT, which is one of the major commercial cloud-IoT platform, and develops an access control model

for virtual objects (shadows) communication in AWS IoT called AWS-IoT-ACMVO. Chapter 6 concludes this dissertation.

Chapter 2: BACKGROUND AND RELATED WORK

In this chapter, we review various areas as follows. We review academic published IoT architectures, which we used to propose our IoT architecture. Also, we review main issues of the IoT that we tried to overcome within our proposed architecture. The main benefit of integrating the cloud layer in the middle layer of the IoT architecture is also discussed. Also, we review general access control models and access control models for IoT environment. Finally, we review the AWS IoT platform and the access control model of AWS IoT.

2.1 IoT Proposed Architectures

Various IoT architectures have been proposed, and these are divided into different layers. The general construction proposed in most IoT architectures includes three basic layers: an object layer, one or more middle layers, and an application layer [1, 8, 58, 74, 76]. In all the papers that we reviewed, an object layer and an application layer existed. Although the functionalities of the object and application layers might vary in their detail, the general functionalities are similar. On the other hand, the middle layers vary in terms of the number of sub-layers and the proposed technologies. We will discuss each layer in the general IoT architecture as well as its entities and functionalities.

2.1.1 The Object Layer

The main task of the object layer (e.g., perception layer [74,81], hardware layer [29]) is to identify objects [74], collect data from the physical environment [1,58], and reconstruct a broad perception of the data. This task is accomplished by using objects (devices) such as sensors that can query location, humidity, temperature, motion, etc. [1].

All papers agree that the primitive entity of this layer is sensors. Some papers would describe this layer as consisting of wireless sensor networks (e.g., cluster of sensors [72]), where sensors are the main physical objects that collect data. Other proposals would add other entities to this

layer, such as actuators [1, 29], RFID tags [58], devices (e.g., cameras and cellphones) [68], and networks of devices [81].

The IoT relies on a pervasive and heterogeneous set of objects that produce big non-structured or semi-structured data [13]. These objects generally have limited computational power and low storage capacity. Since IoT technology is a rich producer of big data [21], which is collected by constrained objects, the collected data needs to be transferred to a more capable layer through secure channels to solve a specific problem. Moreover, with a large set of heterogeneous objects which have different operating conditions, functionalities, resolutions, etc. [68], providing seamless integration of these devices is a huge challenge, and this issue may hinder object interoperability and slow down the improvement of a unified reference model for IoT [52].

2.1.2 The Middle Layers

The main goal of middle layers is to successfully convey the collected data from object layer to a remote destination [76, 81]. Many proposed IoT architectures describe the middle layers as only one layer. A transmission layer (gateway) proposed in [81] is responsible for gathering/sending data, packaging data, exchanging data, parsing/dispatching commands, and logging events between the application and object layer. All data is saved in the application layer in a database. A network layer is proposed in [22, 39, 76] as a middle layer; it is responsible for intelligently processing the massive amount of collected data.

While the transmission layer in [81] and the network layer in [76] are the only layer in the middle layer of IoT architecture, other IoT architectures have proposed two or three layers between the application and the object layers. The proposed architecture in [20] consists of two layers in middle, the network layer and the service layer. The network layer connects everything together to share information, and it aggregates information from existing IT infrastructures such as power grids and healthcare systems. The service layer includes service discovery, service composition, trustworthiness management, and service APIs. The IoT architecture in [41] also introduced the network and middleware layer in the middle. The network layer transmits information to the

middleware layer, which has service management, link to the database, information processing, automatic decision, and a ubiquitous computation unit that can be placed in the cloud.

Separating tasks between a network and a service/middleware layer [20, 41] is more reliable than loading the network layer [22, 39, 76] with so many tasks. The service/middleware layer includes important tasks such as processing received data, managing services, making decisions, and computing tasks. Many papers suggested integrating with the cloud to support tasks in this layer [13, 14, 41, 57, 60].

The main functionality of the middleware layer is providing a common set of device functionalities [1, 13, 24]. The middleware layer can also be divided into sub-layers. It is divided into two sub-layers in [1], which are the object abstraction and the service management layers. The service management layer pairs services of objects with requests for them, processes received data, and makes decisions, while the object abstraction layer transmits data collected by objects to the service management layer. Cloud computing and data management processes are implemented at the object abstraction layer. Other papers have proposed a middleware layer that is divided into three sub-layers: an object abstraction, a service management, and a service composition layer [8, 24]. The service composition layer offers the functionalities for the composition of single services, which are represented at the service management layer.

An approach to integrating cloud computing as a middle layer in the IoT architecture is proposed by Gubbi et al. [29], where the IoT architecture includes three layers: a wireless sensor networks layer, a cloud computing (middle) layer, and an application layer. Integrating cloud in the middle offers various functionalities to support a middleware layer. Gubbi and other researchers [13, 14, 57, 60] have discussed the features of integrating cloud computing with the Internet of of Things.

The object abstraction layer is discussed in many papers, although they offer slightly different definitions. In [8], this layer consolidates access to the heterogeneous devices in the object layer, while in [24], it allows physical objects in the object layer to deliver their capabilities and features to the upper layers. It is also called virtual object layer in [68]. Virtual object is defined in [73]

comprises both current and historical information about a specific physical object. However, Amazon called virtual objects as device shadows which persist both last and desired future status of each device even when the device is offline. The advantages of representing virtual object (e.g., digital counterpart of physical objects) as major component of IoT are discussed in depth in [52].

2.1.3 The Application Layer

Application layer is the top most layer of the IoT architecture, and it delivers services and system functionalities to the end users. The application layer presents the information to users through merged and analyzed data. This layer exploits the functionalities of the middleware layer and provides a rich and user-friendly application of the IoT. Using applications is an easy way to remotely communicate to objects (devices) and present their information. The final received information from the middleware layer can be used to create models, graphs, and flowcharts, which can support decision-making [1, 8, 74].

2.2 IoT Major Issues

The basic idea of the IoT concept is the pervasive presence of a variety of connected objects [8] that implement any possible interaction [52]. The evolution of IoT has encompassed various technologies that assist in the delivery of smart objects, which provide real-world awareness and interactivity [42]. In the following sections, we will review the IoT issues that we recognized.

2.2.1 Devices Heterogeneity

As IoT environment characterized by the pervasive presence of networked objects, different capabilities of computations and communications are presented. For example, different protocols are developed for networking such as ZigBee and Bluetooth, but each protocol has its own specific characteristics. Thus, the management of this heterogeneity is a considerable point in modeling a unified IoT architecture [49, 52].

2.2.2 Scalability

With an expected 50 billion smart objects in existence by 2020, scalability issues arise, such as communication with an enormous number of objects, management of the digital counterparts of objects, and management of an enormous number of service execution options. Thus, management of this very-large-scale system is very demanding [49, 52].

2.2.3 Addressing and Identification

The development of the Internet has led to people being interconnected, but the current trend is leading to objects being interconnected. Thus, all heterogeneous connected objects in the IoT need to be identified with a unique ID that facilitates objects interconnection. This identifier can also facilitate controlling and gathering information from objects. This could be done mainly in two ways. Objects could be identified by physically attached tags such as RFID tag [49], or they may be addressed by Internet protocols such as IPv6 [75]. Lately, solutions have been developed to run in resource-constrained environments such as Message Queue Telemetry Transport (MQTT) [35].

2.2.4 Self-Organization Capability

Once objects have been identified and become networked, it is necessary to communicate additional information about them. Thus, objects need to organize themselves, and the functionalities and services of objects need to be announced. Also, applications need to dynamically discover these functionalities and services. Identified objects can interact with each other without human intervention. However, integrating functionalities and/or services that are provided by objects and dynamically searching for and discovering these functionalities and/or services [25] are issues in an IoT system.

2.2.5 Constrained Resources

Often, objects in the IoT may not be able to implement intensive computational tasks and store big collected data. This is because of limited energy, storage, processing of intensive tasks, etc.

For example, wireless sensors have limited amounts of battery and do not have enough space to implement complex computational tasks. Therefore, objects need to collaborate with another party, such as using the cloud [13, 57], to overcome their limitations.

2.2.6 Mobility

In a wireless context, one reason for complicity is device mobility. In the IoT, objects are very likely to be mobile [82]. Mobility management is essential to provide unified connectivity regardless where the objects are located or moved to. Objects could be moved to a different cell within the same system or it could be moved to a totally different system. All kinds of mobility need to be taken into consideration.

2.2.7 Security and Privacy

Pushing IoT technology into the real world will require overcoming security problems. Since the IoT architecture comprises pervasive heterogeneous objects, big collected data, entities interacting with each other, and applications interacting with the entities, security is one of the most important issues in designing such architecture. Protecting the data and services and controlling the number of critical incidents will affect the entire IoT. However, protecting the IoT is a complicated and challenging task because global connectivity and accessibility are key aspects of the IoT. Many security and privacy issues need to be addressed, such as resilience to attacks, data authentication, access control, and client privacy to address threats such as denial of service (DoS), physical damage, and eavesdropping [61, 82].

2.3 Integration of Cloud and IoT

An approach that integrates cloud computing as a middle layer in the IoT architecture is proposed by Gubbi et al. [29]. Other researchers have described the integration of cloud and the IoT by a specific name, such as the CloudIoT [14, 57]. Many of the papers reviewed have discussed advantages of integrating cloud computing with the Internet of Things, but several other papers see

that the IoT as filling the Cloud's gaps (mostly the limited scope) [14]. We will discuss benefit of this integration as follows.

2.3.1 Communication

Communicating among pervasive objects and applications is one of the key aspects of the IoT. The management of this communication, which can be from anywhere at any time, can be very expensive. The cloud supports effective and economical solutions to connect, track, and manage things from anywhere at any time through customized portals and built-in apps [14, 60]. Thus the cloud can help in delivering managed data and services to the application layer.

Although communication can be improved and simplified by using the cloud, communication problems, such as bottlenecks or connection failures, require specific support to avoid an accumulation of errors. These problems could arise from scenarios where transferring a huge amount of data or continuous transfer of data is required [14].

2.3.2 Storage

IoT is a rich producer of big data [21] that is collected by objects. The data repository for storing such big data will need enormous space to encompass all the possible collected data. Also, the data needs to be organized and analyzed before being delivered to applications. All these tasks can be provided by the cloud, which has a virtually unlimited, low-cost, on-demand storage capacity. It can assist with data aggregation, integration, and preservation. Thus, with the cloud, data can be accessed from anywhere [14].

2.3.3 Computation

Since objects in the IoT have limited energy, storage, processing of intensive tasks, etc., integrating cloud computing, which offers the capacity for the storing, analyzing, computing, and visualization of big collected data [29], with IoT is promising. To make sense of the collected saved data, it is important to develop artificial intelligence algorithms, which can be offered by the cloud. Decision-

making also needs artificial intelligence algorithms to allow aggregation and combination of data as well as to permit accessing to this data. These algorithms are also used in the analyzing and visualizing processes, to support interpretation of the collected data [71].

2.4 Access Control Models

2.4.1 General Access Control Models

Access control is a fundamental service that supports security in various systems. The fundamental access control approaches involve an Access Matrix, which is a conceptual model that indicates the right that each subject possesses for each object. Correspondingly, there is an access control list (ACL) that stores the matrix by columns [31,67], where each object is associated with an ACL. while each subject is associated with a capability list that stores the matrix by rows [31,67].

Other earlier approaches include discretionary access control (DAC) and mandatory access control (MAC) [66, 67]. DAC allows owners to control the access of users to objects. For each request of a user to access objects, the object authorizations are checked to validate whether an authorization that the user can access exists or not. Despite the flexibility of DAC, it is not adequate to enforce information flow policies since copying information from one object to another is not constrained. MAC is an approach that constrains information flow policies by assigning subjects and objects a security level (also called a clearance). Users have no discretion to decide which users are allowed to access specific objects.

Role-based access control (RBAC) model is an access control approach that has been frequently used and continue to be the desired delegated technology. With this approach, an administrator creates roles that present specific tasks; grants permissions to those roles (permission-role assignment), which are difficult to change; and assigns users to those roles as regards their responsibilities (user-role assignment), which can be changed more frequently. Mathematically, roles hierarchies are partial orders, which are reflexive, transitive, and antisymmetrical relations [64].

Provenance-based access control (PBAC) use provenance data to make access control decisions to targets. The provenance data could be related to specific user or object, and it could be data of

transaction records that are captured in a system. [50] identified the possible use of provenance data in classifying traditional Dynamic Separation of Duties (DSOD) issues.

Relationship-based access control (ReBAC) has been widely studied and adapted for access control in online social networks (OSNs). The authorization policies are typically expressed in terms of relationships of the requester and the target [17]. Generally, ReBAC alone is not enough to enforce different security and privacy requirements that meet today systems growth. ReBAC is integrated with attribute-based policies in [18].

Attribute-Based Access Control (ABAC) model is a form of access control that has recently attracted the interest of both academic and industry researchers. The National Institute of Standards and Technology (NIST) recently described a high level access control model that uses attributes [33, 34]. Jin et al [40] have proposed a unified ABAC model that can be configured to the traditional access control models (i.e., DAC, MAC and RBAC). Researchers have also studied combining attributes with RBAC. Kuhn et al [43] presented models that combine ABAC and RBAC in various ways, while Yong et al [78] proposed extending the roles of the RBAC with attributes. Al-Kahtani et al [2] introduced the notion of using attributes in user-role assignment of RBAC model. Thus there has been a tradition of research on combining or relating attributes to various access control models, old and new.

A novel approach to combining attributes with the access matrix was developed by Zhang et al [79], who defined the attribute-based access matrix (ABAM) model by adding attributes to the classic HRU model [31]. In the HRU model each cell $[s_i, o_j]$ of the access matrix contains a set of rights that subject s_i can exercise over object o_j. In general, a subject is also an object while every object is not necessarily a subject. Subjects and objects are collectively called entities. ABAM additionally associates a set of attributes $ATT(o)$ with each entity o. A notable aspect of ABAM is that its commands not only test for and modify rights in access matrix cells like in HRU, but can further test for and modify attribute values. Henceforth we understand ABAM to mean finite ABAM.

The features of attribute testing and modification, also called attribute mutability, were adapted

in ABAM [79] from the earlier UCON model [56]. UCON incorporates various additional features such as ongoing authorization and updates, as well as obligations and conditions. The sub-model of UCON called PreUCON_A [56, 59] where attribute testing and modification are carried out prior to allowing access. Similar to finite ABAM, in finite PreUCON_A the set of attributes is finite and each attribute of an entity can only take on a finite set of permissible values. Henceforth, we understand PreUCON_A to mean finite PreUCON_A.

We have published a study that investigates the theoretical relationship between ABAM and PreUCON_A [5]. Our first observation is that ABAM is an extension of HRU and thereby inherits the undecidable safety results of HRU. On the other hand PreUCON_A is known to have decidable safety analysis [59]. It follows that ABAM cannot be reduced to PreUCON_A. On the other hand, we show how PreUCON_A can be reduced to ABAM. This construction inspires us to define a restricted version of ABAM named RL-ABAM2, which stands for right-less ABAM with two parameters as will be explained. We then prove that PreUCON_A and RL-ABAM2 theoretically have equivalent expressive power.

Our constructions suggest the power of using formulas in PreUCON_A, absence of which in ABAM leads to having to an explosion of ABAM commands in the PreUCON_A to ABAM reduction. Conversely, the ability to activate multiple rights in a single RL-ABAM2 command leads to multiple PreUCON_A commands in the ABAM to PreUCON_A reduction. These features could be combined in a more usable model. Finally, the study of ABAM indicates that a safe application of access rights could be based on the following principles. Firstly, do not use rights in the if part of commands. Secondly, some rights could be left behind by commands so their next use is more efficient. Our comparative study of PreUCON_A and ABAM suggests there is a meaningful place for access matrix rights, even as access control research and practice is tending towards attributes.

2.4.2 Access Control Models for IoT

Several access control models for IoT have been proposed to address security and privacy issues, as surveyed in Ouaddah et al [54]. Using capability-based access control (CAC) model for IoT

has been proposed because entities hold granted rights that support different levels of granularity with possibility of delegation, while similar functionality is not feasible with ACLs. However, the main two major drawbacks of using the capability approach are propagation and revocation [28]. Mahalle et al [47] propose the identity authentication and capability-based access control (IACAC) model, where devices use an access point and the CAC model to be connected to each other. Moreover, in [30], the capability-based access control system (CapBAC) is used in controlling access to services and information. The authors described use cases and argue that CapBAC supports rights delegation, least privileges access principle, more fine-grained access control, fewer security issues, and fewer issues related to complexity and dynamics of subject's identities than ACLs, RBAC and ABAC. In [77], a simple-efficient mutual authentication and secure key establishment based on ECC, which has much lower storage and communication overheads, is proposed for the perception (object) layer of the IoT.

2.4.3 The Publish and Subscribe Communication Paradigm

The publish/subscribe communication interaction scheme is suitable for large-scale distributed interactions such as IoT. It lets subscribers indicate their interest in topics (events) services that manage and deliver data generated by publishers. In other words, producers publish data on a software bus (topic/event service), and consumers subscribe to the information they are interested in receiving from that bus (topic/event).

The publish/subscribe paradigm has various implementation styles [10, 23], primarily topic-based and content-based. The topic-based scheme is similar to the notion of groups where consumers (subscribers) become members of a topic (a group), and producers (publishers) publish data to a topic. All subscribers to that topic are informed about the published data. In contrast, the content-based approach introduces a subscription scheme based on the published content. Here, subscribers specify filters, which define constraints based on the name-value pairs of the published properties (content) and use single or combined basic comparison operators (=, $<$, $\leq$, $>$, $\geq$) to identify events of interest [10, 23].

2.5 The General Access Control Model for AWS-IoT (AWS-IoTAC)

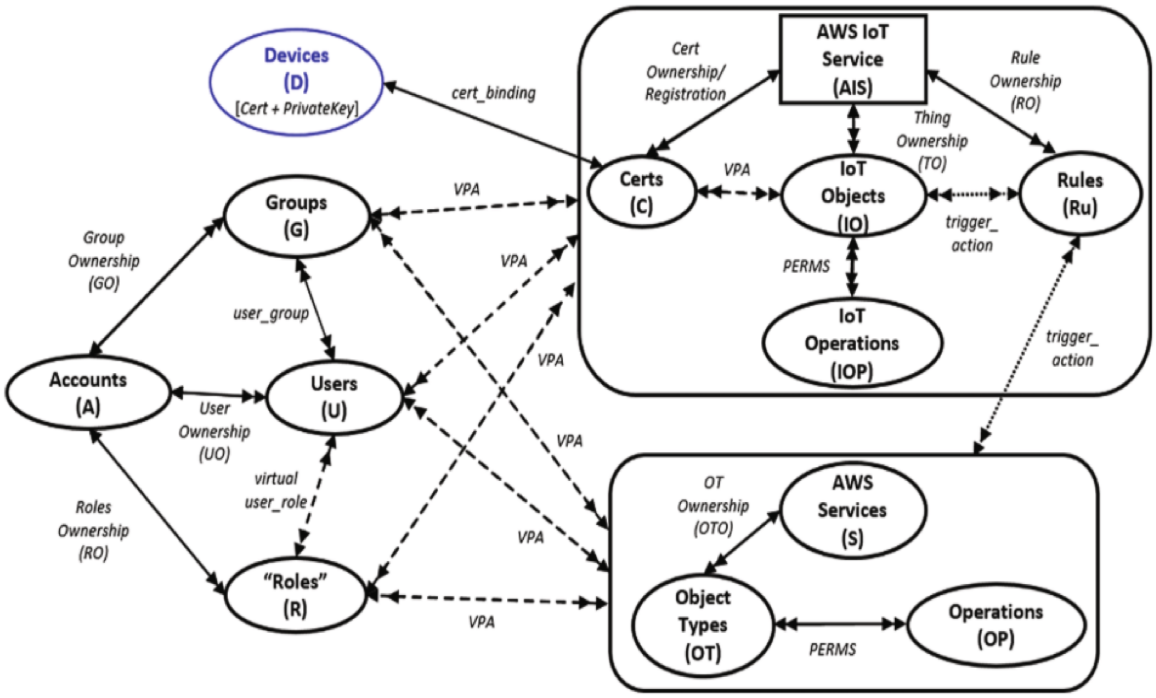

Figure 2.1: AWS IoT Access Control (AWS-IoTAC) Model within a Single Account [12]

Bhatt et al [12] study AWS IoT as a major commercial cloud-IoT platform, and develop a formal access control model for AWS-IoT called AWS-IoTAC. This access control model is an extension of AWS access control (AWSAC) model previously developed by Zhang et al [80] for AWS access control in general.

AWS-IoTAC comprises all the components and relations of AWSAC with modified or extra set of components and relations related to the AWS IoT service. The main component of AWSAC are Accounts (A), Users (U), Groups (G), Roles (R), Services (S), Object Types (OT), and Operations (OP). The additional components in AWS-IoTAC, which are related to AWS IoT service, are Certificates (C), IoT Objects (IO), IoT Operations (IOP), Rules (Ru), Virtual Permission Assignment (VPA), and Devices (D).

Users can access cloud resources through an account when they authenticated by AWS IoT. Within users account, they can give permissions or create new users. Thus, they own their accounts

as an administrator. Users also can be part of one or more than one group. Users can have one or multiple roles to allow them to gain access to corresponding cloud resources (services), which have one or multiple objects type. Objects can have one or more assigned operations, which represent the allowed operations on the object types based on an access control policy that is attached to them. All the previous component already existed within AWSAC.

Within AWS IoT service, the AWS IoT Service (AIS) is appeared to support IoT devices and their underlying authorization in the cloud. Also, certificates can be used by IoT devices or applications to authenticate to AIS and authorized to communicate with IoT Objects. Each certificate is attached with IoT Operations to authorize devices to communicate with the AWS IoT Service. When devices such as sensors are connected with AWS IoT, set of virtual IoT Objects called things or thing shadows can be used to represent them in the cloud even when they are disconnected. The functionality of these entities and their relationship to each other are shown in figure 2.1 and formally described in [12].

Chapter 3: ACCESS CONTROL MODELS FOR CLOUD-ENABLED INTERNET OF THINGS: A PROPOSED ARCHITECTURE AND RESEARCH AGENDA

Acknowledgment: The materials in this chapter are published in the following venue [4]:

- *Asma Alshehri and Ravi Sandhu. Access control models for cloud-enabled internet of things: A proposed architecture and research agenda. In the 2nd IEEE International Conference on Collaboration and Internet Computing (CIC), pages 530-538. IEEE, 2016.*

This chapter describes our proposed access-control oriented (ACO) architecture for cloud-enabled Internet of Things and the characteristics of each layer. This architecture serves as a framework to build access control models for a cloud-enabled IoT.

3.1 Motivation and Scope

With the development of wireless communication systems over the last few decades, the concept of Internet of Things (IoT) has emerged and has recently attracted increasing attention of governments, companies, and academia. The IoT is an extension of network technology, where the basic core of communication is the Internet. The promising IoT paradigm integrates many widely dispersed, mobile, abundant, heterogeneous objects, such as sensors and actuators that collect data from an environment and in turn act upon it. Many industries have initiated major projects even in the absence of widely accepted architectures for IoT. Thus, there is a crucial need to develop consensus architectures for the future IoT.

There have been various proposals for IoT architecture in the research literature. The proposed IoT architectures can be divided into three main layers: an object layer, one or more middle layers, and an application layer. The main difference in detail between them is in the middle layers. Some architectures abstracted the middle layers to only one layer [76], while others have two or more middle layers [1, 8, 58, 74].

Integrating the cloud as a central entity is suggested in various IoT architecture [1,8,58,61,74]. The IoT can gain advantage from the powerful capabilities and resources of the cloud to offset its technological constraints. The IoT encompasses pervasive and heterogeneous objects that produce big non-structured or semi-structured data. IoT objects have limited computational power and low storage. Offering virtually unlimited computational capabilities, low-cost, on-demand storage capacity, and ubiquitous resources usable from everywhere, the cloud is the most convenient and cost-effective solution to deal with IoT technological constraints [13, 14, 57, 60].

Moreover, several research papers have suggested incorporating an object abstraction layer as an essential part of IoT architecture. Atzoori et al. [8] argue for such a layer to unite access to the heterogeneous devices in the object layer. Evangelos et al. [24] suggest the benefit of exposing the capabilities and services of objects to the upper layers through such abstractions. A similar definition as [24] is given in [68] with the name of 'virtual object layer'. A virtual object is also described in [73] as comprised of both current and historical information about a specific physical object. A virtual object is called a device shadow in Amazon Web Service (AWS) IoT, which persists the final and desired future status of each device, even when the device is offline. The potential benefits of using virtual objects are discussed in depth in [24], with respect to IoT issues such as scalability, heterogeneity, security and privacy, and identification.

In this chapter, we take a first step toward our eventual goal of developing an authoritative family of access control models for a cloud-enabled Internet of Things. We build upon previously published IoT architectures, which are all roughly divided into three layers: an object layer, one or more middle layers, and an application layer. In the different approaches the middle layer is divided into sub-layers differently. Since several papers discuss the advantages of using the cloud and virtual objects to solve IoT issues, our proposed access-control oriented (ACO) architecture supports using them in the middle of object and application layers. As a result, our proposed architecture is divided into four layers: an object layer, a virtual object layer, a cloud layer, and an application layer. This 4-layer architecture will be our guide to build access control models for a cloud-enabled Internet of Things. Within this architecture, we present several illustrative examples

that expose some IoT access control issues. This leads us to discuss needed access control research to address these issues.

In the following sections, we propose a cloud-based IoT architecture and its characteristics in Section 3.2. Illustrative examples are discussed in Section 3.3. A research agenda for access control based on our proposed architecture is discussed in Section 3.4.

3.2 Access Control Oriented Architecture for IoT

From the reviewed papers, we conclude that there is a need for integrating the cloud with the IoT architecture and a benefit to using virtual objects as a counterpart of physical objects. Therefore, we propose an IoT architecture that emphasizes enabling cloud computing to support middleware and service management functionalities. Our architecture is designed to assist in proposing access control (AC) models for IoT, and thus we call it an AC-oriented (ACO) architecture for the IoT. We will show the details of this architecture and examples of it in this section.

Our ACO architecture is designed to be roughly close to the general architecture of the IoT that is divided into three layers: the application layer, one or more middle layers, and the object layer. Therefore, we kept the two basic layers (the application and object layers) that exist in all of the reviewed IoT architectures. However, the middle is divided into two layers: a virtual object layer and a cloud services layer. Therefore, our architecture basically includes four main layers: an object layer, a virtual object layer, a cloud services layer, and an application layer. Figure 3.1 represents the layering of ACO architecture for the IoT where object layer appears at the bottom and application layer at the top. Each layer has its components and functionalities. We discuss each layer below.

3.2.1 The Object Layer

This layer is similar to most of the object layers that we reviewed. The main task of this layer is to collect data from physical environment and to construct a broad overview of the data to send it to the upper level (virtual object layer) or to other objects. This layer includes heterogeneous types

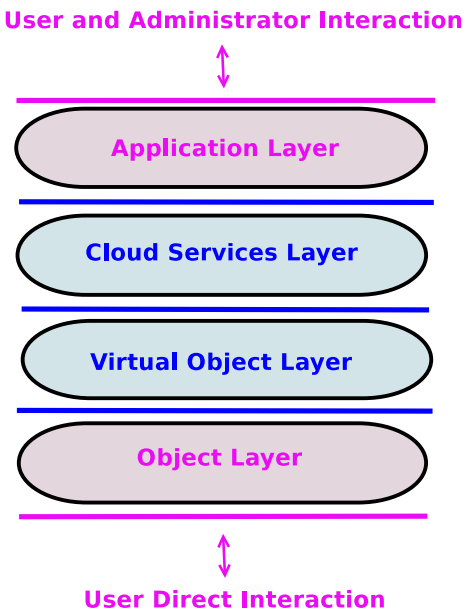

Figure 3.1: ACO Architecture for the Cloud-Enabled IoT

of objects such as sensors, actuators, and cameras, which form one cluster or multiple clusters of objects.

Objects at this level basically push collected data to upper layers, such as data collected from sensors. However, objects at this layer also can receive information from other objects or from higher layers. For example, a light bulb needs to receive a command to be turned off or on. Thus, data at object layer could be output of objects or input to them.

The object layer is on the bottom of the IoT architecture. Users can directly communicate with objects by pressing a button, changing a device, powering on an object, etc. Objects in this layer can communicate with other objects directly through communication technologies such as Bluetooth, Wireless, ZigBee, 6LoWPAN, ISA 100, WirelessHart/802.15.4, 18000-7, and LTE [1]. They can also communicate to their virtual objects (digital counterparts) through the Internet. In both communication directions, there is a need to authenticate the communication possibly using technologies such as public key infrastructure (PKI) or digital certificates.

The physically connected objects could be intentionally or unintentionally turned off or on. At the same time, the input/output data of physical objects could be needed/reached any time. Thus, knowing the status of objects in IoT architecture is required. One way to do it is to have

virtual objects of physical objects. In addition, most of the physical objects will have limited computational power and low storage, and can only implement simple computational tasks and save limited data. Since IoT relies on vast sets of collected data, these physical objects need to depend on another party to execute intensive computational tasks, as well as to voluminous collected data. This party will be the cloud services layer, which will be described in Section 3.2.3.

3.2.2 The Virtual Object Layer

In the virtual object layer, virtual objects (digital counterparts) can present a persistent current status of objects if both are connected. In case the virtual and physical objects are not connected, virtual objects could also present a desired future status, the last received status of a physical object, or both the future and last received status. Virtual objects deliver the services and capabilities of physical objects to users. Virtual objects can have a subset of physical objects' services, all of the physical objects' services, or one of physical objects' services. In our model, we will assume virtual objects only for physical objects. There is no digital counterpart for users (although that may be appropriate as the architecture and functionality evolve).

Using virtual objects solves IoT issues such as scalability, heterogeneity, security and privacy, and identification. Thus, the virtual objects in this layer can uniformly communicate with each other regardless of heterogeneity and locality in the object layer. This communication needs to be controlled by appropriate access control mechanisms, such as RBAC [64] or ABAC [40], or ReBAC [17]. Studying the benefit of using multiple access control mechanisms is also possible [5].

Virtual objects can be associated with physical objects in various ways. The simplest is to represent one virtual object with one physical object (if any) that has one or many services, thus leading to a one (or less)-to-one association [44]. With an object that has many services, there is also the possibility of representing one virtual object for each service, thus leading to a one-to-many association. For example, a smartphone could represent all of its services through a single virtual object (one-to-one), or it could have separate virtual objects one for each available service, e.g. one for location sensing and one for temperature sensing, thus resulting in a one-to-many

association [37, 52]. Another way would be to represent a set of physical objects with one virtual object, for instance, to manage them more efficiently with less resource consumption than having a distributed implementation (many-to-one) [52, 62], or to collect the information of single service from various physical objects (many-to-one) [52, 69]. Thus, the combination of all different kinds of associations will lead to many-to-many association [38].

3.2.3 The Cloud Services Layer

This layer is built to assist most of the functionalities related to the service/middleware layer. With an expected 50 billion smart objects in existence by 2020, attention must be focused on developing the means to access, store, and process the huge amount of data collected by these objects. Thus, this layer assists in storing and processing the big collected data. The saved data in this layer can also be used intelligently for smart monitoring and actuation, and it can be visualized in ways that are more meaningful for users. Thus, policymakers (or administrators) can utilize the visualized data to help them to modify or add policies that are kept in the cloud, so the communication and access between applications and objects are managed through the cloud. The cloud services layer also assists in the intensive computational tasks that cannot be handled by the constrained objects. Thus, the cloud services layer supports the computation, visualization, and analysis of stored data in the cloud.

In addition to managing the communication with applications and objects, clouds can also communicate with each other, ranging from only providing services and information at a local level to collaborating with other connected IoTs in order to share information at a broad level and pursue common goals. Hence, multi-cloud communication can occur at this layer. As we mentioned above, controlling accessing to data and entities communications can be controlled by suitable access control mechanisms such as RBAC [64], ABAC [40], and ReBAC [17].

3.2.4 The Application Layer

The application layer is the top most layer of the proposed ACO IoT architecture and offers an interface through which users can easily communicate with objects and visualize the analyzed information. Administrators can also interact with applications to generate policies or to update/add policies based on the obtained information. Moreover, configuring and managing the communication of objects and virtual objects is organized by administrators through applications. General users and administrators can remotely communicate with IoT objects and virtual objects only through applications. For example, a user who is out of her home can use an application to send a turn off command to remote light bulbs located at her home. Applications communication with any entity should be controlled and authorized by using appropriate access control techniques.

3.3 Illustrative Example

The proposed ACO architecture for the IoT in the previous section is illustrated in more concrete terms in Figure 3.2 in context of a simple example.

3.3.1 Multi-Value Switch Use Case

In our example, we have a multi-value switch that can change the color of a light bulb to red, blue, or green. Users communicate directly (and physically) with the multi-value switch to turn on the light bulb with a specific color. We will discuss each layer as follows.

The Object Layer

Although our ACO architecture in general allows objects to communicate directly to each other, we don't allow that in our example for simplicity. Therefore, the multi-value switch and the light bulb (objects) do not communicate with each other directly at this layer. Both of the multi-value switch and the light bulb connect to the Internet via secure channels to communicate with their virtual objects. Thus, the only communication allowed is with the virtual object layer.

Figure 3.2 shows one multi-value switch (object) that enables a color changing service. In

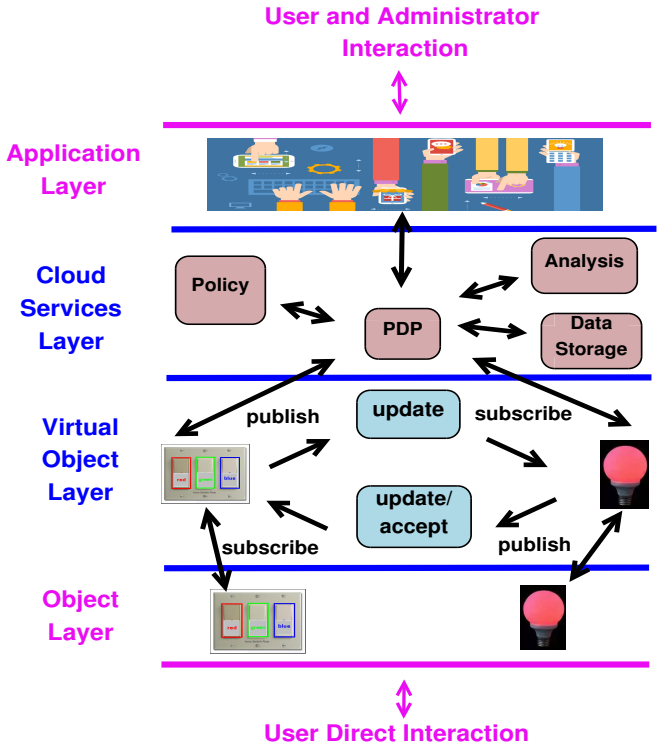

Figure 3.2: Multi-Value Switch Use Case for the ACO Architecture

other words, we have a physical object that has one service. Therefore, there is one virtual object that can associate with each multi-value switch, leading to a one-to-one association with the virtual object. Also, there is only one light bulb that receives a command to change its color, and for that light bulb there is one associated virtual object.

Users can directly interact with the light bulb and the multi-value switch by powering them on or off, changing them, or moving them, etc. Also, users can interact with multi-value switch by pressing a color. In our example, the collected data is only coming from users' action. When users press a color in the multi-value switch, the command is sent to virtual objects. The light bulb also needs to communicate with its virtual object to receive the new color; otherwise the color will stay the same. Over time, collected commands and received colors from the multi-value switch generate data that can be logged and saved.

The Virtual Object Layer

The virtual multi-value switch and virtual light bulb (virtual objects) store information about their corresponding physical objects. The virtual multi-value switch saves the current pressed command in the multi-value switch if they both are connected, and it will save the last received command in case they are not connected. Similarly, the virtual light bulb will maintain the current color of the light bulb if they both are connected, and it will also save the last received color (the future color of the light bulb once it is connected) in case they are not connected. The current, past, and future status can be presented via list of attributes (e.g.'current-status', 'past-status', and 'future-status') that are saved in the virtual objects.

The two virtual objects communicate in three different ways. They communicate with their physical objects. They also communicate with each other directly at this layer. One familiar communication model between virtual objects is publish/subscribe [7]. Our simple use case has two topics: 'update' and 'update/accept'. The virtual multi-value switch publishes to 'update' topic, and the virtual light bulb subscribes to 'update' topic and thus receives any published command to change the state of the color; and vice versa with 'update/accept' topic. Finally, virtual objects can communicate with the cloud services layer to log and save the sending commands and the received colors, store the number of disconnections with physical objects, share attributes with the policy decision point (PDP), and receive authorized topics to publish or subscribe, and so on.

The Cloud Services Layer

This layer supports cloud services such as compute, storage and analysis of stored data. As shown in Figure 3.2, the cloud services layer has data storage that saves all the collected data (as discussed above). This data can be analyzed and visualized to decision makers to understand, for example, the difference between the number of sent commands from multi-value switch and the number of received commands to light-bulb.

The policy component stores rules that allow virtual objects to publish/subscribe to 'update' or 'update/accept' topics. In our example, the light bulb is allowed to publish to 'update/accept' topic

but not to publish to 'update' topic. Policy rules are constructed and managed by administrators who communicate to this layer through applications. The PDP communicates with policy and data storage components, and with virtual objects to retrieve required information (e.g. roles and attributes) for making a decision [34]. For instance, it decides whether or not users can communicate to virtual objects and thus objects themselves.

The Application Layer

This layer includes applications to view the analyzed and visualized saved information, such as past statuses of the multi-value switch and the light bulb. The applications allow the owner of the multi-value switch and light bulb to control communication between virtual objects, users' access to the saved data, and communication between applications and objects by constructing policies that are used by the PDP, which control various kinds of communication.

3.3.2 Multi-Value Switch Use Case Enhancements

Our use case showed a very simple scenario that has only two objects. Each object has one-to-one association with its virtual objects. This example can be enhanced in several ways such as adding multi-value switches and virtual objects, allowing direct communication between switches and light bulbs, or permitting collaborative multi-clouds, etc. Some examples of the enhancements are discussed as below.

As the number of rooms increase, more light bulbs are needed, and thus using one multi-value switch can control all of them efficiently. Introducing more light bulbs that connect to one virtual light bulb leads to a many-to-one association on the light bulbs side, which helps to manage them more efficiently, while there is a one-to-one association on the multi-value switch side. Figure 3.3(a) shows how one multi-value switch can control many light bulbs. However, how to control each room with different color is not clear with a many-to-one association on the light bulb side.

On the other hand, looking to control many rooms separately will result in increasing the num-

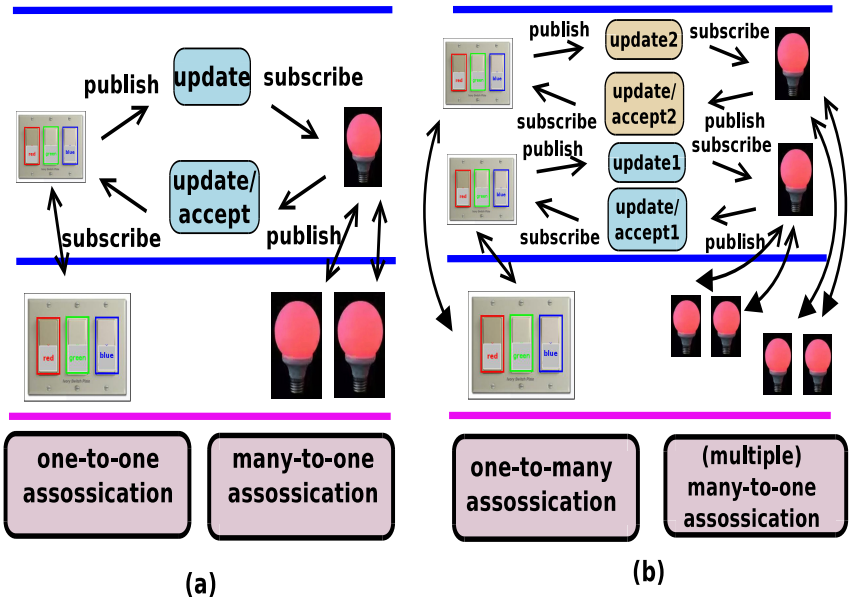

Figure 3.3: Different Kinds of Objects and Virtual Objects Associations

ber of multi-value switches, virtual multi-value switches, and virtual light bulbs. Designing a smart multi-value switch that considers how many times the red, the green, or the blue button has been pressed can result in a one-to-many association. In other words, this smart multi-value switch is associated to many virtual objects, rather than having multiple multi-value switches, each of them is associated with one virtual object. Figure 3.3(b) shows that with two groups of light bulbs (two rooms), each group associates with a different virtual light bulb (multiple many-to-one associations). The smart multi-value switch will be associated with two virtual multi-value switches. The first virtual multi-value switch is for the first group of light bulbs, and the other one is for the second group. Hence, two many-to-one associations are on the light bulbs side, while one-to-many association is on the smart multi-value switch side, which decreases the cost of having many multi-value switches.

In our simple case example, there is a one-to-one association on both the light bulb and the multi-value switch. One virtual multi-value switch is communicating with one virtual light bulb by pushing to ‘update1’ topic (this update is only for specific virtual object(s)). As a result, there are only two topics to publish and subscribe: ‘update1’ and ‘update/accept’ topics. However, Figure 3.3(b) shows more topics since we are looking to control two separate groups of light bulbs.

Thus the 'update1' topic is for the first group and 'update2' is for the second one. In this case, adding a third group of light bulbs to be controlled separately will increase the number of topics.

One advanced enhancement is having groups of light bulbs and multi-value switches in one city, each group is for one neighborhood. The logged data, such as historical multi-value switch commands, is saved in the cloud. Another city that has a different cloud would like to communicate with the first city's cloud and retrieve the analyzed historical multi-value switch commands to discover the most required color in that city, for example, or to study the difference between the number of sent commands from multi-value switch and the number of received commands to light-bulb, and so on. This case shows why different clouds could communicate and collaborate within the cloud services layer in multi-cloud collaboration.

In the application layer, a smart phone could have an application that displays for users the current color, the past color, and the future color of a light bulb, as well as an illustrative graph that visualizes the number of times each color has been pressed so that users can understand what the most desired color has been. In addition to multi-value switch commands, an application could allow users to control the light bulb color remotely by pressing the required color and transmitting it within the cloud and the virtual object layer.

3.3.3 Controlling Communications and Data Access

Various access control models have been discussed such as attribute based access control model (ABAC) [40], relationship based access control model (ReBAC) [17], and role based access control model (RBAC) [64]. Access control models such as these can be employed to control communications between entities and controlling accessing to data.

In our simple use case, we can control virtual objects communication by adapting an appropriate access control model. ABAC model, for example, shows its capability for accommodating the need of the IoT in terms of the unlimited increase of objects. ABAC can be used to control communication between our two virtual objects: the virtual multi-value switch (VO1) and the virtual light bulb (VO2). Controlling which virtual objects are authorized to publish or subscribe to

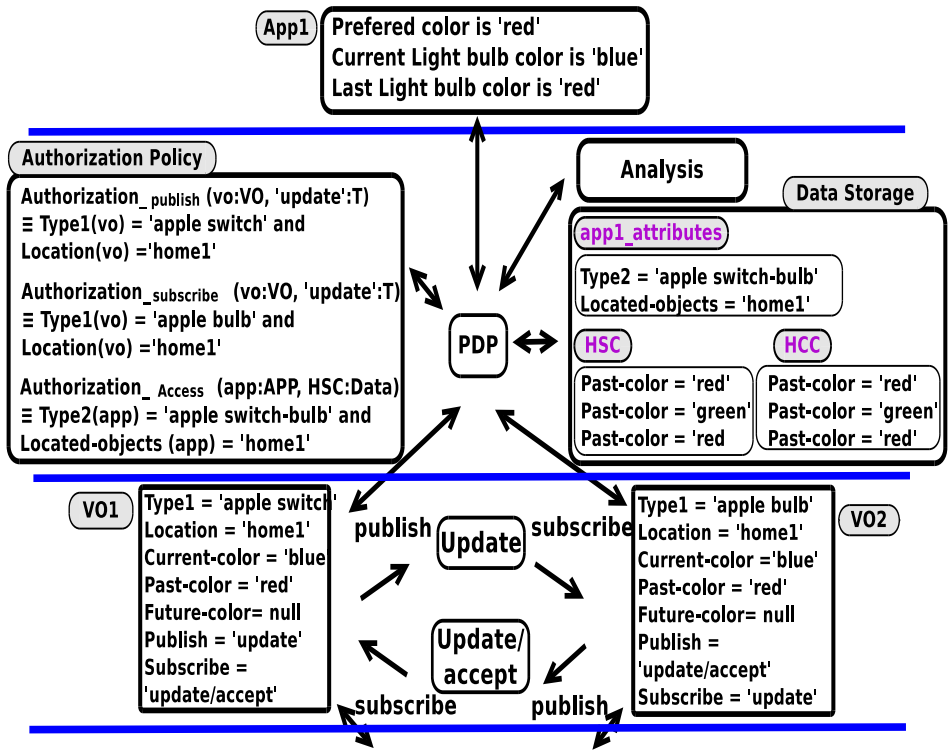

Figure 3.4: Using ABAC to Control Virtual Objects Communications

a specific topic is important here. In our case, the VO1 needs to be authorized to publish to the 'update' topic and subscribe to the 'update/accept' topic, while the VO2 needs to be authorized to subscribe to the 'update' topic and publish to the 'update/accept' topic.

For both of the two virtual objects, we have the following attributes: {Type1, Location, Current-color, Past-color, Future-color, Publish, Subscribe}. Each attribute has different range, so the range of each attribute is as following: range: range(Type1) = {'apple switch', 'apple light bulb'}, range(Location) = {'home1'}, range(Current-color) = range(Past-color) = range(Future-color) = {'red', 'green', 'blue'}, and range(Publish) = range(Subscribe) = {'update', 'update/accept'}. We assume that Type1 and Location attributes' values are already assigned for both of the two virtual objects. Thus, a virtual object is allowed to either publish to 'update' topic if it is with Type1 'apple switch' and is located at 'home1', or it is allowed to subscribe to 'update' topic if it is with Type1 'apple light bulb' and is located at 'home1', and vice versa for the 'update/accept' topic. Figure 3.4 shows the authorization policy to publish or subscribe to 'update' topic and VO1 and VO2 attributes.

Historical sent commands (HSC) from VO1 and historical changed colors (HCC) of VO2 can be logged in the cloud storage. In that case, access control techniques are needed to control accessing to historical data. For example, an application (App1), which represents information about historical sent commands and received colors, needs to access data storage to get that information. This application has the following attributes: {Type2, Located-objects}, and their ranges are as follows: range(Type2) = {'apple switch-bulb'}, range(Located-objects) = {'home1', 'home2'}. We assume that the application is already identified and the Type2 and Located-objects attributes values are already assigned for the application. By using ABAC model, applications can access historical sent commands and historical changed colors only if they have the following attributes values: Type2 = 'apple switch-bulb' and Located-objects = 'home1'. Figure 3.4 shows the authorization policy that allow an application to access historical data and application saved attributes. Also, it shows information that can be presented via the application (App1).

3.3.4 Object Life Cycle Issues

Looking to object layer in our simple use case, we have two objects that need to be designed with at least basic requirements of Internet of Things objects. For example, multi-value switches that do not connect to the Internet and communicate with virtual objects are not eligible to be placed with the Internet of Things objects. Thus, objects need to be designed and built to communicate.

Objects need to hold identifiers to be recognized once they connect to the Internet. With object identification, each object could be mapped to their virtual objects and authorized for communication with virtual objects. In our example, we have the virtual multi-value switch and the virtual light bulb. The decision of mapping light bulb to virtual multi-value switch or to virtual light bulb will need the light bulb identifier. Therefore, identifying objects is one important aspect of the object layer.

For each object there is at least one owner responsible for configuring that object, controlling its communication to other objects, and authorizing users and application to control/connect to this object. In other words, owners are the only users who can manage object policies. The multi-value

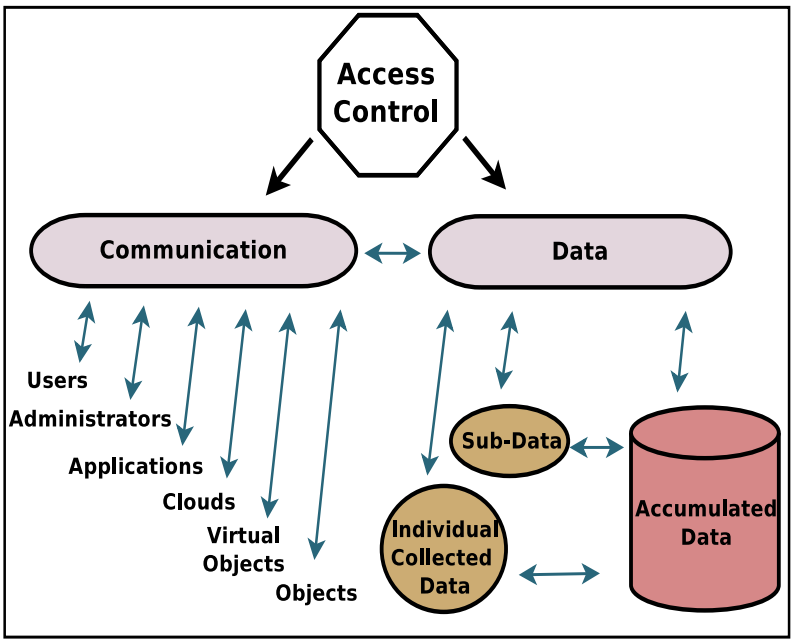

Figure 3.5: Recognized Access Control Issues within ACO Architecture

switch in our example is permitted to send a command color only via owner authorization policy. As a result, we can say that ownership is significant for object security.

Designing objects to communicate with the assistance of identifiers and owner guidance leads to the secure deployment of these objects [36]. The secure communication of objects needs to be maintained periodically for these objects. The light bulb, for example, needs to be checked frequently for whether it is still permitted to communicate with a virtual light bulb or not. Additionally, objects or virtual objects that are not working any more or are not needed need to be changed or removed. Thus, ownership and policies of retired objects should be revoked for security purposes.

3.4 Research Agenda

Our use case reveals different possibilities of communications among entities in each layer and in different layers. As a result of these communications, the collected data flows among entities in various layers. From our ACO architecture and illustrative examples, we recognize two major issues that need to be controlled: communications among entities, and data that flows through these

communications. Figure 3.5 represents the general two main recognized issues and entities in each of them. As an initial step toward understanding and introducing access control models for virtual objects communication, a simple example of using ABAC model to control a publish/subscribe communication style between virtual objects is discussed above in Section 3.3.3.

General Issues may also appear and need to be handled. For example, the existence priority of physical objects and their virtual objects is one issue. it is important to address questions such as whether virtual objects exist first or physical objects?. Also, administrator and users both access through applications, so it is important to distinguish administrators from users. Questions such as " How administrators control communications between entities at same and different layers?", "What kinds of actions that are self-control?", and "What kinds of actions need direct control from administrators?" need to be considered.

Chapter 4: ACCESS CONTROL MODELS FOR VIRTUAL OBJECT COMMUNICATION IN CLOUD-ENABLED IOT

Acknowledgment: The materials in this chapter are published in the following venue [6]

- *Asma Alshehri and Ravi Sandhu. Access control models for virtual object communication in cloud-enabled IoT. In Proceedings of the18th International Conference on Information Reuse and Integration (IRI). IEEE, 2017.*

This chapter proposes and discusses operational and administrative access control models for virtual object communication.

4.1 Motivation and Scope

The concept of the Internet of things (IoT) originated from the evolution of wireless communication systems over the last few decades. The technologies of sensing, networking, software architectures, information management, data analytics, and visualization all converge in IoT. The IoT gives rise to new security challenges that call for a significant revision of current security solutions, including access control systems. In the prior chapter we have developed an access-control oriented architecture (ACO) [4] for cloud-enabled IoT, comprising four layers: an object layer, a virtual object (VO) layer, a cloud services layer, and an application layer (see Section 3.2). ACO architecture recognizes the need for communication control within each layer and across adjacent layers, coupled with the need for data access control at the cloud services and application layers. In this chapter, we focus on developing access control models for VO communication, within the developed ACO architecture.

A virtual object can communicate in various ways. The current common method is topic-based publish-subscribe scheme (see Section 2.4.3). For instance, a virtual object is called a device shadow in Amazon Web Service (AWS) IoT. The device shadows service uses reserved MQTT [53] topics to permit applications and things to get, update, or delete the state information

for a device [70] by publishing and subscribing to MQTT topics.

The traditional access control models are access control lists (ACLs), capability lists, and role-based access control (RBAC). Attribute-based access control (ABAC) is receiving current interest as a more general model that encompasses the benefits of prior traditional models, as well as brings new features suitable for dynamic and open environments such as the IoT. In this chapter, we develop access control models for VO communication in two layers: operational models and administrative models, assuming topic-based publish-subscribe interaction among VOs. Operational models are developed using (i) ACLs for topics and capabilities for virtual objects, and (ii) ABAC. It is argued that RBAC is not suitable for this purpose. Administrative models for these two operational models are developed using (i) ACLs, (ii) RBAC and (iii) ABAC. A use case illustrates the details of these access control models for VO communication, and their differences. To the best of our knowledge, this is the first work to address control of VO communication in the IoT.

The rest of this chapter is organized as follows. In Section 4.2, we introduce a use case about sensing the speed of cars, and flagging those above the limit. In Section 4.3, we propose and discuss appropriate operational access control models for virtual object communication. Section 4.4 discusses administrative models for this purpose. Assessments of our models with respect to the IoT security and privacy preserving objectives, which are proposed in [54], are discussed in Section 4.5.

4.2 Use Case within ACO Architecture

We employ a use case of sensing speeding cars, illustrated in Figure 4.1, as a running example. A car is declared to be speeding if two sensors within a specified distance sense the speed to be over limit. Such cars will be reported along with a picture. Each car is assumed to have an RFID that enables sensors to identify cars. Objects in physical object layer (sensors and camera) generally have limited computational power and low storage. Also, issues such as scalability, heterogeneity, security and privacy, and identification can be handled easier within virtual object layer than object layer. Thus, we assume that the sensors and camera push collected data (e.g.

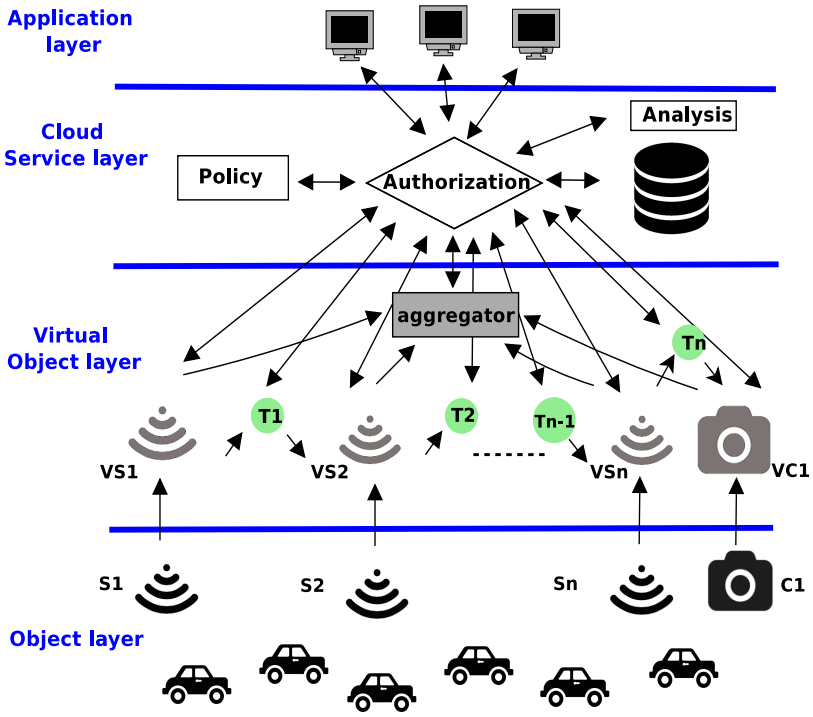

Figure 4.1: Sensing Speeding Cars within ACO Architecture

RFID) to their virtual objects where more powerful computations could happen. For this use case, we assume communication between sensors can occur only within the virtual object layer, via publish/subscribe to topics, and they cannot communicate directly with each other.

The physical objects are sensors $S1, ..., Sn$ and a camera $C1$. Correspondingly, in the virtual object layer, there is a group of virtual sensors $VS1, ..., VSn$, one for each physical sensor, and one virtual camera $VC1$ for the physical camera. The physical sensors are linearly arranged along the highway. Similarly, their virtual sensors communicate in a linear sequence. We have topics $T1, ..., Tn$-1, where $T1$ enables communication from $VS1$ to $VS2$, and so on, through Tn-1 which enables communication from VSn-1 to VSn. Finally, topic Tn facilitates communication from the last virtual sensor VSn to the virtual camera $VC1$.

Physical sensors have the capability to sense speed and RFID of cars at the location where the sensor is located. Moreover, they have local storage and simple computation capability for the collected data to be refined and pushed to their VOs. The physical camera has the capability to sense RFID, current location, and take pictures. It has local storage, and local simple computation

for the collected data to be refined and pushed to VC1. However, if VOs are not connected to physical objects, the collected data will be kept temporarily in the local memory of physical objects. Eventually, the refined collected data will be pushed to the VOs and will be removed from local memory.

The scenario of communication among virtual objects starts with $VS1$, which publishes a suspicious RFIDs list of over-limit cars, received from $S1$, to $VS2$ through $T1$. $VS2$ also receives a suspicious RFIDs list from $S2$. $VS2$ compares these two suspicious RFIDs lists. RFIDs that occur on both lists are added to a SavePic list located on $VS2$ as well as pushed to an aggregator, which is responsible to consolidate all incoming data from VSs and VC1 and push it to storage in the cloud services layer. RFIDs which occur on only one of the lists, are consolidated in a suspicious RFIDs list at $VS2$. Then, $VS2$ publishes the SavePic list and the suspicious RFIDs list to the next virtual sensor $VS3$. Other sensors and virtual sensors perform similar steps. $VC1$ compares the RFIDs (along with pictures) coming from $C1$ with RFIDs on the SavePic list. The matched RFIDs will be pushed, along with the taken pictures to the aggregator. Note that a picture is taken of over-limit car by the camera $C1$ and communicated to $VC1$, but only pictures of cars that are in the SavePic list are sent to the aggregator and communicated outside the VO layer. The other pictures are discarded. This shows the privacy benefit of separating the VO layer with transient information from the cloud services layer with persistent information.

4.3 Operational Access Control for VO Communication

We develop access control models for VO communication in two layers. The operational model specifies controls regarding which VO to VO communications are authorized via topics. The administration of these controls is specified by the administrative models. This separation of operational and administrative models was first introduced in role-based access control, where operational models were defined in [26, 64] and administrative models in [63].

In this section we develop two operational models: ACLs and capabilities-based access control, and attribute-based access control. Publish/subscribe schemes typically employ message brokers

(MBs) [10] (also called event brokers [23]) that route messages from publisher to subscribers for topics. After subscribers register (by sending a subscribe request) with a message broker of a topic, a published message to that topic will be forwarded by the message broker to all subscribers. The operational authorizations specify which VOs are allowed to publish to which topics, and likewise which VOs can subscribe to which topics. These authorizations determine the permitted pattern of communication in the VO layer, and thereby indirectly in the object layer.

The operational access control models address the following questions. Which VOs are allowed to publish or send a subscription request to a topic's MB? Which VOs should a topics MB forward data to? Which MBs should VOs publish to or send a subscription request to? Which MBs should VOs receive data from? These lead to the following related questions. Where should the publish and subscribe controls be located? On the topic side, virtual object side, or both?

The operational models recognize two sets of entities: virtual objects (VO) and topics (T), and a set of rights R={p,s} denoting publish and subscribe respectively. VOs are active entities that can publish data to topics, and receive data from topics they are subscribed to. Each topic has an associated MB, which responds to subscribe requests from VOs, accepts data published to the topic by a VO and forwards this data to the topic's subscribers. The right for the forward operation is represented in the singleton set F={Forward}. Note that these entities are very different in nature from the usual user/subject and resource/object entities in access control models [40, 64].

4.3.1 ACL and Capability Based (ACL-Cap) Operational Model

The ACL-Cap model incorporates ACLs for topics and capability lists (Cap) for VOs, as illustrated in Figure 4.2. These lists are maintained by administrators, as will be discussed in Section 4.4. The ACL of a topic comprises a list of VOs, along with a publish or subscribe right for each VO. The capability list of a VO similarly comprises a list of topics with the publish or subscribe right for each topic. The capability list informs the VO as to which topics it can publish or subscribe, obviating the need for additional logic for this purpose. Similarly, the ACL of a topic informs the topic's MB as to which VOs can publish or subscribe to it. Since the VOs and topic MBs are fully

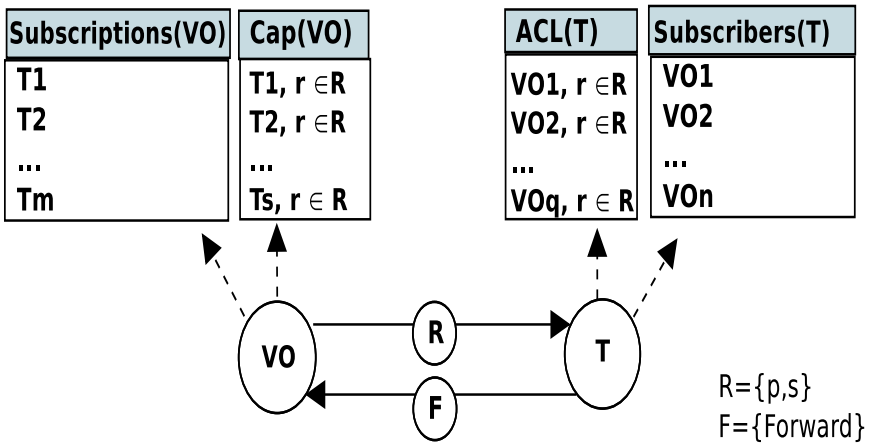

Figure 4.2: The ACL-Cap Model

automated, this dual ACL-Cap approach is more convenient and secure relative to ACL-only or capability-only approaches. This dual scheme allows unauthorized operations to be denied at the earliest possible moment, instead of deferring the decisions till later.

A particular VO can publish to topics for which it has a publish capability. The publish operation will succeed only if that topic's ACL has a corresponding entry for that VO with the publish right. The authorization rule for publish is therefore expressed as follows.

$$Auth\text{-}Publish(VO, T) \equiv (T, p) \in Cap(VO) \wedge (VO, p) \in ACL(T) \tag{4.1}$$

The subscribe operation is more complicated, in that the subscribe relationship needs to be established before published data is forwarded and received. This requires a request to subscribe from a VO to a topic, and an accepting response from the topic's MB. Recognizing that this is a multi-step operation, we express the authorization rule for successful completion of subscribe as follows.

$$Auth\text{-}Subscribe(VO, T) \equiv (T, s) \in Cap(VO) \wedge (VO, s) \in ACL(T) \tag{4.2}$$

A successful subscribe operation adds the topic T to the VO's subscriber list, and the VO to the topic's subscriber list as shown in Figure 4.2. The actual forwarding of published data by a topic's

Table 4.1: ACL of Topics

$T1$	....	Tn-1	Tn
$VS1$, p	....	VSn-1, p	VSn, p
$VS2$, s	....	VSn, s	$VC1$, s

Table 4.2: Capability List of VOs

$VS1$	....	VSm	$VC1$
$T1$, p	....	Tn, p	Tn, s
	....	Tn-1, s	

MB to a VO is authorized as follows.

$$Auth\text{-}Forward(T, VO) \equiv VO \in Subscribers(T) \wedge T \in Subscriptions(VO) \tag{4.3}$$

Equations 4.1 and 4.2 respectively address the questions: which VOs are allowed to publish or send a subscription request to a topic's MB? Equation 4.3 addresses the question as to which VOs a topic's MB can forward data to. Note that equation 4.1 can be partially checked at the publishing VO's side, thus preventing a rogue VO from indiscriminately attempting to publish to unauthorized topics (as would be possible in an ACL-only approach).

For the use case defined in Section 4.2, we have $VO = \{VS1, .., VSn, VC1\}$ and $T = \{T1, .., Tn-1, Tn\}$, with the ACLs and capability lists given in Table 4.1 and Table 4.2.

4.3.2 ABAC Operational Model

Next we develop an ABAC operational model, illustrated in Figure 4.3. The entities in this model are the set VO of virtual objects and the set T of topics with rights R={p,s} and F={Forward}, as before. We have a set of attributes, VOA for virtual object attributes and TA for topic attributes, as follows.

$$VOA = \{VO\text{-}Publish, VO\text{-}Subscribe, VO\text{-}Subscriptions, VO\text{-}Location\}$$

$$TA = \{T\text{-}Publish, T\text{-}Subscribe, T\text{-}Subscribers, T\text{-}Location\}$$

The T-location and VO-location attributes are atomic valued and give the location of the corresponding topic and VO in appropriate units. The remaining attributes are set-valued. Values for VO-Publish, VO-Subscribe, and VO-Subscriptions are a subset of the topics T. Values for T-Publish, T-Subscribe, and T-Subscribers are a subset of the virtual objects VO. The following authorization rules express the same policy as in Section 4.3.1.

$$Auth\text{-}Publish(VO, T) \equiv T \in VO\text{-}Publish(VO) \wedge VO \in T\text{-}Publish(T) \tag{4.4}$$

$$Auth\text{-}Subscribe(VO, T) \equiv T \in VO\text{-}Subscribe(VO) \wedge VO \in T\text{-}Subscribe(T) \tag{4.5}$$

$$Auth\text{-}Forward(T, VO) \equiv T \in VO\text{-}Subscriptions(VO) \wedge VO \in T\text{-}Subscribers(T) \tag{4.6}$$

The attributes VO-Publish, VO-Subscribe, T-Publish and T-Subscribe are assigned by administrators. The VO-Subscriptions and T-Subscribers attributes are assigned as a consequence of establishing the subscribe relationship as discussed in Section 4.3.1. The T-location and VO-location attributes are enhancements to the use case in the ABAC model. We assume that VO-Location is automatically assigned to be the location received from the physical sensor. The T-location attribute is assigned by an administrator. We can conjunctively add the following condition to each of the three equations above.

$$T\text{-}Location(T) \approx VO\text{-}Location(VO) \tag{4.7}$$

This will further constrain the pattern of communication amongst the VOs by taking their location into account. In particular, if sensors are moved a significant distance the authorized communication will be disrupted. Small movements will be accommodated due to the approximate matching in this condition.

Note that a single ABAC authorization rule incorporates topic and virtual object attributes.

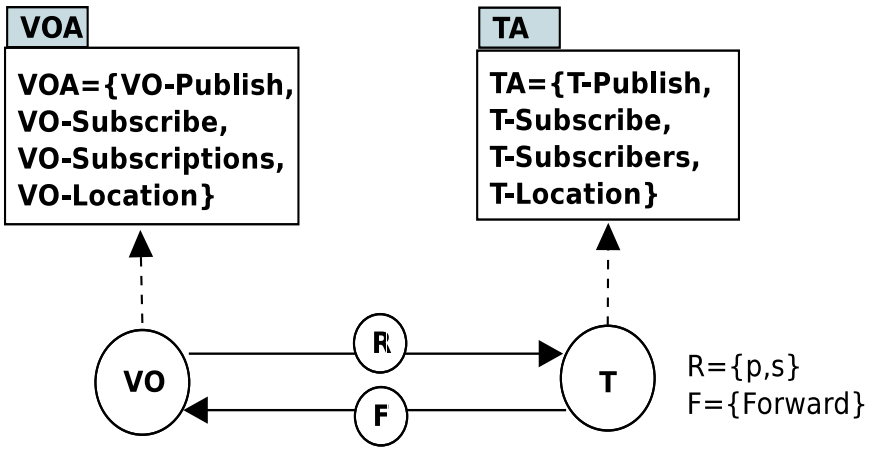

Figure 4.3: ABAC Operational Model

In this respect equations 4.4, 4.5 and 4.6, are respectively similar to equations 4.1, 4.2 and 4.3. However, ABAC allows incorporation of additional attributes such as in equation 4.7, whereas the ACL-Cap model is limited to the ACL and Cap lists as the only permitted "attributes."

4.3.3 RBAC Limitations

In closing this section we discuss some limitations of RBAC in context of IoT VO communications. RBAC was invented with the notion of assigning users to roles, through which users acquire permissions primarily to perform operations on target objects. Virtual objects and topics do not fit this paradigm very cleanly. Virtual objects are active entities in regard to publish and subscribe operations, while they are targets for forward operations. Similarly, topics are targets with respect to publish and actors with respect to forwarding and accepting subscribe requests. The active aspects of virtual objects and topics can be accommodated in RBAC by assigning these entities to mutually exclusive sets of roles. With respect to equations 4.1 and 4.2, the first part of the equations (i.e., $(T, p) \in Cap(VO)$ and $(T, s) \in Cap(VO)$) could be represented by permission assignment of topic permissions to the VO's role. The second part (i.e., $(VO, p) \in ACL(T)$ and $(VO, s) \in ACL(T)$) could similarly be represented by permission assignment of virtual objects permissions to T's role. But this splits the equations into separate roles, which must thereby both be considered when access decisions are made. This consideration of roles of both actor and target

requires major extension to conventional RBAC [64].

4.4 Administrative Access Control for VO Communication

In this section, we develop three administrative access control models to control VO communication, respectively using ACL, RBAC and ABAC approaches. An administrative model is an essential complement to the operational models described earlier. At the same time the structure of an administrative model is not tightly coupled with that of the operational model, as will demonstrate. We use the terms admins to mean users who are authorized to control VO communication, by adjusting configuration of the operational model. For simplicity, we assume admins of topics are same as admins of related VOs.

For the ACL-Cap operational model we have two main administrative questions: Who is allowed to add or delete (VO,p) or (VO,s) from ACL of T? Who is allowed to add or delete (T,p) or (T,s) from Capability list of VO? Slightly different administrative questions arise for the ABAC operational model: Who is allowed to assign or delete values to/from attributes of T? Who is allowed to assign or delete values to/from attributes of VO?

4.4.1 Administrative ACL Model

In addition to the operational model for our use case, the administrative ACL model introduces a set of admin users (A) and admin permissions (AP) as follows.

$$A = \{U1, .., Um\text{-}1, Um\}$$
$$AP = \{Own, Control\}$$

The administrative ACL model has one ACL for each T and VO as shown in Figure 4.4. Own and Control are similar in authorizing modifications to ACLs, Capability lists, and administered attributes of topics and virtual objects as appropriate for the underlying operational model. The difference between Own and Control is that Own authorizes the admin user to grant Own or Control over the topic or virtual object to other admin users, while Control does not.

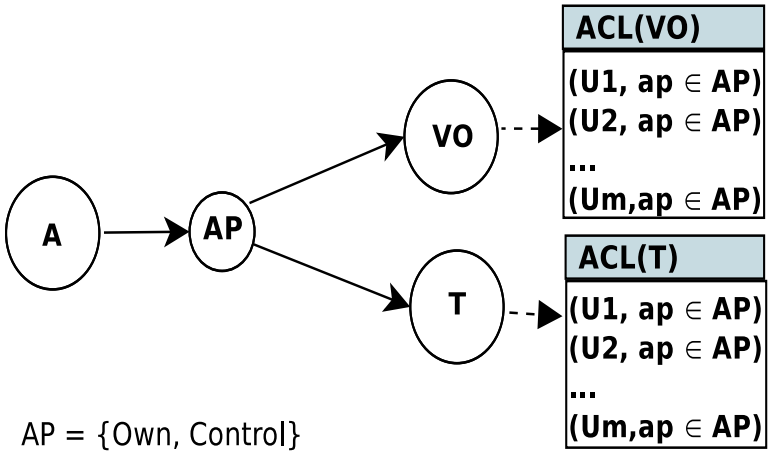

Figure 4.4: Administrative ACL

Table 4.3: All Admins have Own Permission for all VO and T

T1, VS1 Admins	T2, VS2 Admins	.....	Tn, VSn Admins	VC1 Admins
(U1, Own)	(U1, Own)	.....	(U1, Own)	(U1, Own)
....	....	.....	....	....
(Um, Own)	(Um, Own)	.....	(Um, Own)	(Um, Own)

Table 4.4: Only U1 has Own Permission

T1, VS1 Admins	T2, VS2 Admins	.....	Tn, VSn Admins	VC1 Admins
(U1, Own)	(U1, Own)	.....	(U1, Own)	(U1, Own)
(U2, Control)	(U2, Control)	.....	(U2, Control)	(U4, Control)
(U3, Control)	(U3, Control)	.....	(U3, Control)	

A particular admin user U can control T or VO only if (U, ap) is respectively in the ACL of T or VO, where ap is Own or Control. We express the authorization rule for U to control T or VO as follow.

$$Auth\text{-}Control(U, T) \equiv (U, ap) \in ACL(T) \tag{4.8}$$

$$Auth\text{-}Control(U, VO) \equiv (U, ap) \in ACL(VO) \tag{4.9}$$

Administrative ACL Model for Operational ACL-Cap

This model is shown in two different configurations in Tables 4.3 and 4.4. In Table 4.3 all admin users have the Own permission for all topics and virtual objects, while in Table 4.4 only U1 does. Presumably U1 has granted U2 and U3 control over topics T1 to Tn-1, and virtual sensors VS1 to VSn. Control over VC1 is granted to admin U4.

Administrative ACL Model for Operational ABAC

The ACL administrative model does not change structurally, but the meaning of Own and Control are adapted to the ABAC operational model. The difference between Own and Control remains as discussed above, and only impacts the administrative ACLs. For operational ABAC the Control permission over a topic or virtual object authorizes the admin to correspondingly modify topic or virtual object attributes, that are administrable. These are VO-Publish, VO-Subscribe, T-Publish, T-Subscribe and T-Location in our use case. VO-Subscriptions, T-Subscribers and VO-Location are automatically assigned and not administered by admins.

In both cases above, the administrative ACL model has one ACL for each topic and each virtual object. Thus, with large sizes of VO and T, the number of ACL will be larger, and this will be difficult to maintain.

4.4.2 Administrative RBAC Model

The administrative RBAC model for our use case continues to use the set of admin users $A = \{U1, .., Um\text{-}1, Um\}$ and admin permissions $AP = \{Own, Control\}$, introduced in Section 4.4.1. Additionally, it introduces a set of administrative roles (AR) and admin permissions set (APS) as follows.

$$AR = \{AR1, .., ARs\}$$
$$APS = \{(VO \times AP) \cup (T \times AP)\}$$

A particular U can control a topic or virtual object only if U has admin assignment (AA) with

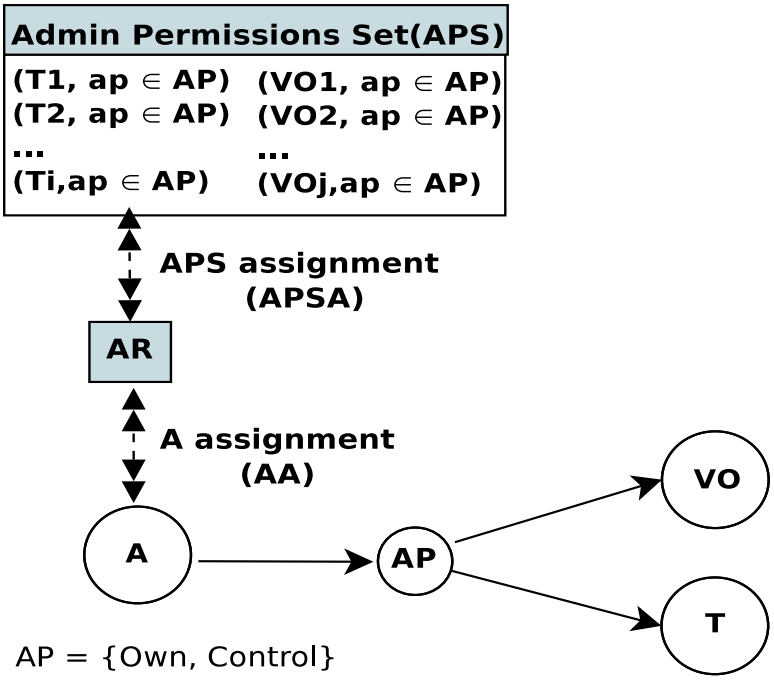

Figure 4.5: Administrative RBAC

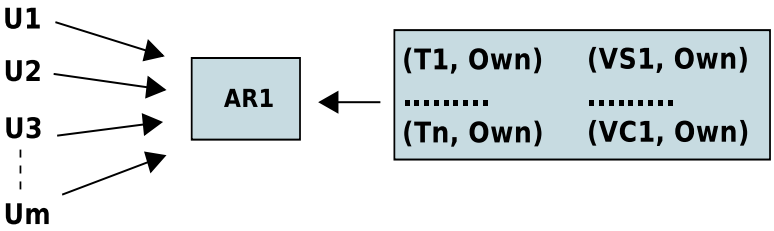

Figure 4.6: Administrative RBAC: Reflects Table 4.3

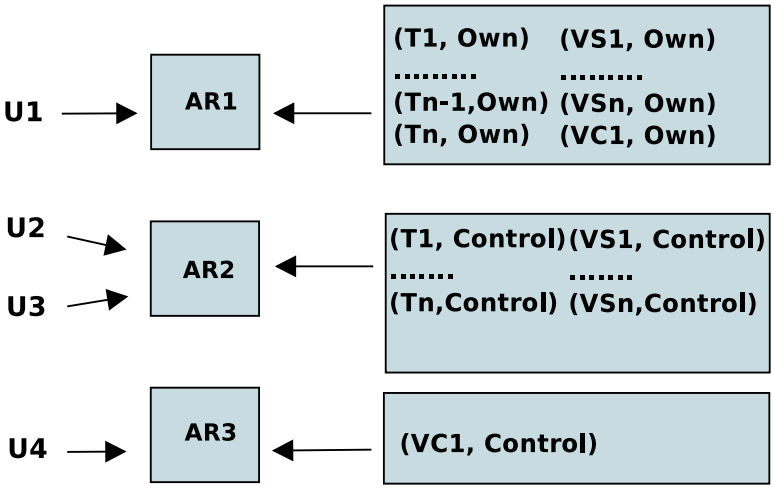

Figure 4.7: Administrative RBAC: Reflects Table 4.4

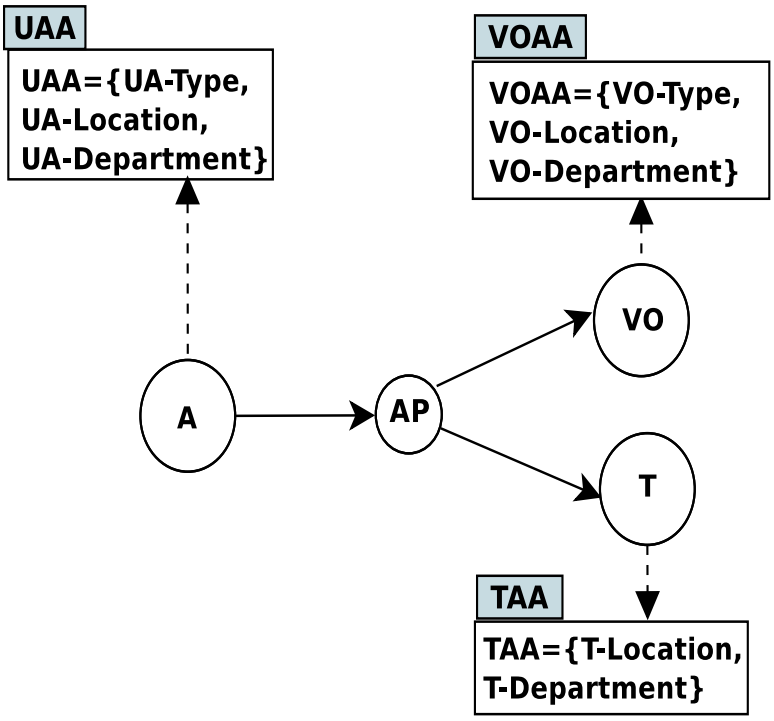

Figure 4.8: Administrative ABAC

some administrative role AR_1 where AR_1 has admin permission assignment set ($APAS$) with that virtual object or topic, as shown in Figure 4.5.

The administrative RBAC model is much easier to maintain than administrative ACL, due to well-known advantages of RBAC over per-topic and per-VO ACLs. The number of administrative roles that need to be managed is reduced to one for the configuration of Table 4.3 as shown in Figure 4.6, and to three for the configuration of Table 4.4 as shown in Figure 4.7. These are constants numbers as opposed to the linear increase in ACLs with increase in topics and virtual objects.

4.4.3 Administrative ABAC Model

The administrative ABAC model for our use case continues to use the set of admin users $A = \{U1, .., Um\text{-}1, Um\}$ and admin permissions $AP = \{Own, Control\}$, introduced in Section 4.4.1. It also introduces administrative attributes for topics (TAA), VOs (VOAA), and users (UAA), as follows.

$$TAA = \{T\text{-}Location, T\text{-}Department\}$$
$$VOAA = \{VO\text{-}Type, VO\text{-}Location, VO\text{-}Department\}$$
$$UAA = \{UA\text{-}Type, UA\text{-}Location, UA\text{-}Department\}$$

Note that these reuse the operational attribute introduced in the operational ABAC model of Section 4.3.2 for Location, and add additional administrative attributes Type and Department. These administrative attributes are atomic valued. The range of the Type and Department attributes are some small number of enumerated items in each case. Figure 4.8 shows TAA, VOAA, and UAA being used to authorize AP for A. The interpretation of the Own and Control permissions for the two operational models is as discussed in Section 4.4.1.

The authorization to use the Control permission with respect to virtual objects or topics is specified as follows.

$$Auth\text{-}Control(U, VO) \equiv$$
$$(UA\text{-}Type(U) = Own \vee UA\text{-}Type(U) = Control) \wedge$$
$$UA\text{-}Department(U) = VO\text{-}Department(VO) \wedge$$
$$(VO\text{-}type = sensor \vee VO\text{-}type = camera) \wedge$$
$$UA\text{-}location \approx VO\text{-}Location(VO)$$

$$Auth\text{-}Control(U, T) \equiv$$
$$(UA\text{-}Type(U) = Own \vee U\text{-}Type(U) = Control) \wedge$$
$$UA\text{-}Department(U) = T\text{-}Department(T) \wedge$$
$$UA\text{-}location = T\text{-}Location(T)$$

These representative equations provide the $Control$ permission to a user who has Own or Control type for a VO if they are in the same Department and approximate Location, provided the VO is of type sensor or camera. The $Control$ permission to a user who has Own or Control type for a topic T is provided if they are in the same Department and exact Location (recall Location of a topic is an administered attribute), provided the VO is of type sensor or camera. In ABAC these rules can be easily modified or refined, e.g., we could have separate rules for sensors and

cameras. ABAC is flexible, scalable and adaptable because it abstracts identity, role, and resources information of ACL and RBAC approaches into VO, topic and user attributes. Also, collected data (e.g. VO-Location) can be used as attributes values, which collaborate with other attributes to make a decision.

4.5 Assessments With Respect to IoT Security and Privacy Objectives

Ouaddah et al [54] discuss security and privacy requirements for several IoT application domains, and classify these into six categories: privacy, technological constraints, social and economic aspects of the IoT, confidentiality and integrity, reliability and availability, and usability. Each category has various objectives. In the following, we will assess these objectives with respect to our ACO architecture, access control models, and our use case.

The privacy of users in our ACO architecture and access control models is generally maintained. The VO layer collects data from objects, so no third parties (such as other virtual objects) can get these data without rights (publish, subscribe) that are managed by admin (admin-driven permissions [54]). Once a right is obtained, VOs can share their data with each other without any intervention (decentralization [54]) or observation by a third party (a user, the Cloud, etc). Our use case identifies over-speed cars and tracks them using RFIDs (pseudonymity [54]), and the collected data are kept in virtual object layer until a decision is made. When a speeding decision is made, pictures of the cars (the cars' license plates are disclosed) are persistently saved along with their RFIDs and shared with the Cloud service layer. Otherwise they are discarded (privacy). Moreover, our ACO architecture helps users to control their collected data by keeping their data in VO layer or pushing anonymous sub-data to the Cloud service layer. Furthermore, access control models help users to control their own data by setting rights for VO communication (user-driven [54]). However, in our use case, the driver of a car has no control over the collected data about their car speed. We also needed to link the specific actions of the same car to track their speed, so the RFID is maintained and a picture is taken if needed.

IoT area has significant technological constraints. Our ACO architecture allows pervasive het-

erogeneous objects (sensors, camera, etc). Once they connect with their VOs, those VOs can communicate through the publish/subscribe scheme despite their heterogeneity. Our access control models for VO communication achieve most of the complex computation within the VO layer, and the physical objects layer is typically collecting data (sensors) and doing very simple computations (if a car is over the speed limit, then push its RFID to VSj).

Our ACO architecture is designed to allow for Cloud collaboration among different organizations. Further, communication among VOs within different organizations is possible using our access control models. For example, collaboration among the ACL of topics and the capability list of VOs models can support interoperability and cooperation by communicating directly (publish to current Cap(VOs), and accepting publish commands from current ACL(T)). In addition, by using ABAC, attributes of VO and T can be shared among the Cloud service layer and decisions made. Moreover, ABAC supports making decisions by using assigned attributes and the surrounding contextual attributes (sensor location, time, etc.) about an object, the environment, or a user (i.e., context awareness [54]) .

The ABAC operational model has a finer level of granularity than the ACL-Cap model, where the specification of the access control rules has more flexibility and incorporates more information about objects, users, and the environment. A VO's access to a topic can be easily revoked by deleting the VO from the ACL of a topic T, a topic T from the capability list of a VO, a VO from T-Publish/T-Subscribe, or a VO from VO-Publish/VO-Subscribe. Finally, admin users with their own permission can grant or delegate their granted permission to other users, and admin users with their own permission can revoke the granted or delegated permission to other users.

Once a right r is granted to VO or T, an access control decision is made regardless of the connectivity of resource owner (i.e., offline mode [54]). The availability time to publish/subscribe to T is ready once ACL of T is checked. That might take little bit more time with ABAC since both of VO and T attributes need to be checked.

The ACL of T and the capability list of VO is easily managed and modified by authorized users. However, because of the need of context awareness in access control decisions, ABAC

facilitates managing access control authorization (Auth-Publish, Auth-Subscribe) by combining various attributes that are related to an object (e.g. Location), the environment (e.g. Time), or the user (e.g. Department).

Chapter 5: ACCESS CONTROL MODEL FOR VIRTUAL OBJECTS (SHADOWS) COMMUNICATION FOR AWS INTERNET OF THINGS

Acknowledgment: The materials in this chapter are published in the following venue [3]

- *Asma Alshehri, James Benson, Farhan Patwa, and Ravi Sandhu. Access control model for virtual objects (shadows) communication for AWS internet of things. In Proceedings of the Eighth ACM on Conference on Data and Application Security and Privacy. ACM, 2018.*

This chapter reconciles the academic access control models that we proposed in chapter 4 with the AWS IoT that is considered as one of the major commercial cloud-IoT platforms. We first develop an access control model for virtual objects communication for AWS IoT. Then, we discuss issues within AWS IoT and possible enhancements.

5.1 Motivation and Scope

The Internet of Things (IoT) raises new security challenges, which require significant revisions and enhancements of existing security solutions, including access control systems. In chapter 3 we developed the access-control oriented architecture (ACO) [4] for cloud-enabled IoT, which includes four layers: an object layer, a virtual object (VO) layer, a cloud services layer, and an application layer. The ACO recognizes the need for communication control within each layer and across adjacent layers, as well as the need for data access control at the cloud services and application layers. Multiple and diverse access control models are required at various points in this architecture, which must collectively enforce access control policies reflecting the complexity of cloud-enabled IoT.

Also in Chapter 4, a set of access control models for VO communications has been proposed and discussed, referred to as ACO-IoT-ACMsVO. These models are developed in two layers: operational models and administrative models. The ACO-IoT-ACMsVO models are illustrated by the use case of sensing speeding cars as shown in Figure 4.1, which is simplified as shown in

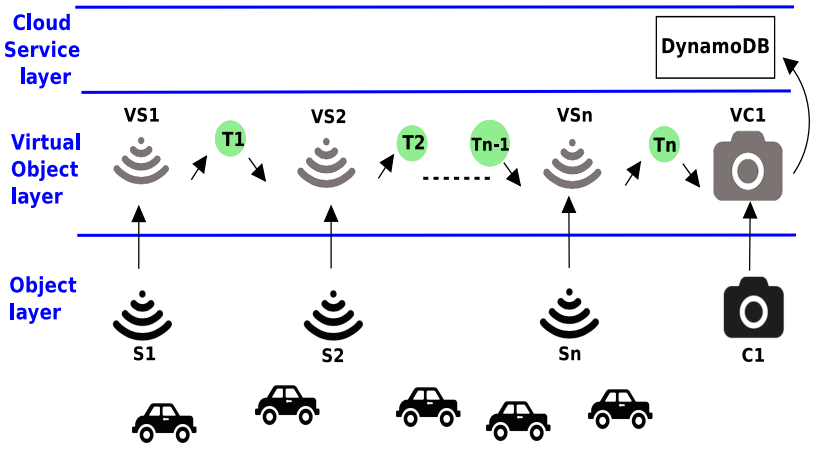

Figure 5.1: The Sensing Speeding Cars Use Case within ACO Architecture [6]

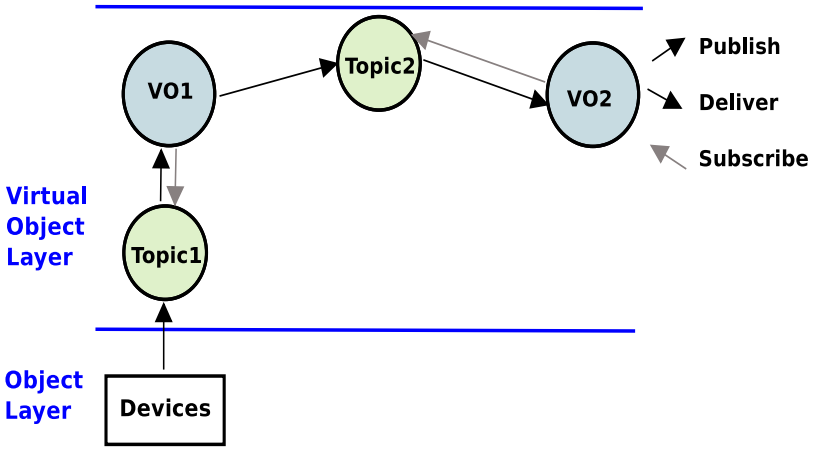

Figure 5.2: The Publish/Subscribe Topic-Based Scheme in the ACO-IoT-ACMsVO

Figure 5.1. This use case will be used in this chapter to implement within AWS IoT.

The ACO-IoT-ACMsVO models are developed utilizing publish/subscribe communication interaction scheme.This scheme is appropriate for large-scale distributed interactions such as the IoT. The basic implementation style of publish/subscribe paradigm is topic-based scheme. The topic-based scheme is comparable to the idea of groups, where producers (publishers) publish data to a topic and consumers (subscribers) become members of a topic (a group) [10,23]. Figure 5.2 shows the general idea of publish/subscribe topic-based scheme that is used in ACO-IoT-ACMsVO.

The principal goal of this chapter is to reconcile the afore-mentioned academic models with a major commercial cloud-IoT platform, viz., AWS IoT. While AWS IoT has a notion of digital shadows closely analogous to VOs, it lacks explicit capability for VO communication and thereby

for VO communication control. Thus, there is a significant mismatch between AWS IoT and these academic models. Nevertheless, as we will show, it is possible to use AWS IoT mechanisms to effectively realize and control VO communications. This demonstrates on one hand that academic models developed independent of AWS IoT can be enforced using this commercially significant platform. It also suggests enhancements to AWS IoT that would be beneficial to facilitate such enforcement. We believe that in the rapidly developing ecosystems of cloud, IoT and their intersection, it is crucial to place academic work within major industry developments. This is the primary motivation for this chapter.

The rest of the chapter is organized as follows. Within AWS IoT, we develop an access control model for virtual object communication (AWS-IoT-ACMVO) in Section 5.2. Section 5.3 discusses the use case of ACO-IoT-ACMsVO within the AWS-IoT-ACMVO model. Section 5.4 discusses proof-of-concept implementations of the use case in two scenarios in AWS IoT platform. Selected performance aspects of our implementation are described in Section 5.5. A discussion of some issues of AWS IoT and possible enhancements are explained in Section 5.6.

5.2 The AWS-IoT-ACMVO Model for AWS IoT Shadows Communication

In this section, based on our extensive exploration of AWS IoT platform, its documentation, and our implemented use cases, we propose an access control model for virtual objects (shadows) communication called AWS-IoT-ACMVO as an abstracted view of AWS IoT capabilities. Figure 5.3 shows the major components of this model, viz., certificates, AWS IoT policies, virtual objects (device shadows), Message Queuing Telemetry Transport (MQTT) topics, and rules engine and its actions. The details of their functionalities are discussed below.

AWS IoT uses X.509 certificates as an identity credential for devices authentication [12]. Certificates can be either an AWS IoT generated certificate or a certificate signed by a AWS IoT registered external certification authority. Generally, one certificate can be given to many **devices**, but it is recommended that each device has a unique certificate to enable fine-grained device management. Figure 5.3 shows that each certificate can be given to more than one device, and each

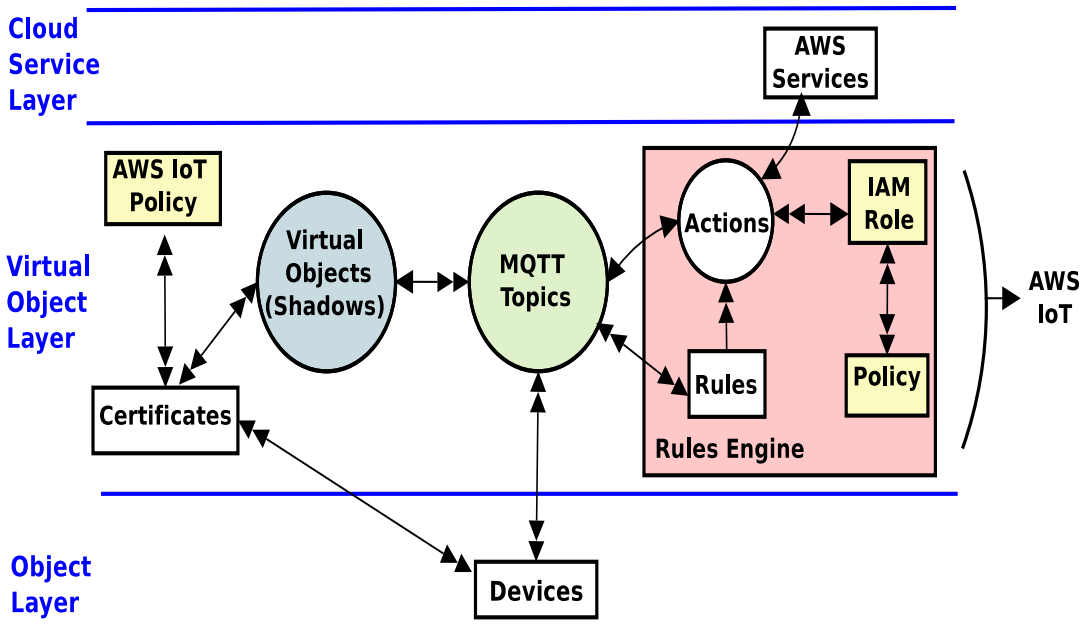

Figure 5.3: The Components of AWS-IoT-ACMVO

device can have multiple certificates (the arrow with double end means a multiplicity). However, every time a device connects it can only activate one certificate.

Once a certificate is generated, there are two AWS IoT entities that need to be attached to the certificate in order to authenticate and authorize AWS IoT devices, which desire to communicate with virtual objects (device shadows), viz., **AWS IoT policy** and **virtual objects**. An AWS IoT policy is a JSON document that is attached to a certificate for authorization purpose. It comprises one or more policy statements, each of which specifies effect, action, resources, and optional condition. An action is an operation that can be granted or denied to a resource as determined by the effect value. Actions can be MQTT policy actions or thing shadow policy actions. The MQTT policy actions are the operations that deal with connecting, sending, or receiving data, which are iot:Connect, iot:Publish, iot:Subscribe, and iot:Receive. On the other hand, thing shadow policy actions deal with permissions to handle virtual objects (device shadows), which are iot:DeleteThingShadow, iot:GetThingShadow, and iot:UpdateThingShadow. Figure 5.3 shows that each AWS IoT policy can be attached to more than one certificate, and each certificate can attach multiple AWS IoT policies. Generally, the AWS IoT policy is attached to a certificate to authorize any kind of actions (e.g. iot:Publish and iot:GetThingShadow) to devices that hold that certificate

(and its private key).

Virtual objects (device shadows) also need to be attached to a certificate as a resource that a device is fully or partially authorized to access. A virtual object can be given more than one certificate, and a certificate can attach to more than one device. Figure 5.3 shows the many-to-many relationship between certificates and virtual objects. A virtual object is also a JSON document that stores information about the current state of a connected device and the desired future state of the connected device (there is no recent or historical state). One of the benefits of the device shadow is that its information can be used to set or get the state of its device, even if the device is not connected. In general, a device that holds a certificate with attached policies and virtual objects has the rights to communicate and access to the attached virtual objects (one virtual object at each connection) based on the attached policies.

In AWS IoT, applications cannot directly update or retrieve data of devices. Virtual objects in AWS work as an intermediate point of communication among applications and physical devices. The only way for applications or devices to interact with a virtual object is to communicate with its **MQTT topics**. In other words, MQTT topics of a virtual object allow applications and devices to get, update, or delete the state information of the virtual object (device shadow) by publishing or subscribing to its MQTT topics. The name of each MQTT topic begin with $\$aws/things/thingName/shadow/\#$, where the thingName is the name of a virtual object, and the symbol $\#$ could be one of the thingName MQTT topics that can be used to interact with the thingName. There are reserved thingName MQTT topics for each virtual object that can be used to publish or subscribe to the virtual object. In order to send a request to a thingName (a virtual object), we only can use /update, /get, or /delete as thingName MQTT topics of that thingName. While /update/accepted, /update/reject, /update/delta, /update/documents, /get/accepted, /get/rejected, /delete/accepted, and /delete/rejected MQTT topics are used by the thingName itself to publish an acknowledgement about accepting or rejecting the received request. Generally, AWS IoT service generates reserved MQTT topics for each created virtual object. The reserved MQTT topics is the only way to communicate with the created virtual object. Figure 5.3 shows that each

virtual object has specific reserved MQTT topics, and each reserved MQTT topic is only related to one virtual object. Moreover, each device can communicate with more than one MQTT topic as long as it has an authorized certificate, and each MQTT topic can be used by more than one device if devices are authorized.

A powerful mechanism in AWS IoT is that a message sent to an MQTT topic can be recognized and analyzed by a rule. **Rules** provide processing for the arrived messages to MQTT topics and enable interactions with various AWS services. A rule consists of a rule name, optional description, SQL statement, SQL version, and one or more actions. The SQL statement is used to filter received messages to MQTT topics, and then the rule engine forwards it to AWS services or republishes it to other MQTT topics by using the action field specified in the rule. There are fixed AWS actions that can be selected, such as inserting a message into a DynamoDB table, invoking a Lambda function, and republishing messages to AWS IoT topics. Thus, rules that are attached to MQTT topics provide ways for virtual objects to interact with AWS services or republish the received messages to other MQTT topics (reserved or unreserved). Figure 5.3 shows that each rule can be triggered by more than one topic, and each topic can trigger more than one rule. Also, when a rule is triggered, one or multiple actions can be executed.

When rules forward the published messages to another AWS service, such as AWS Lambda, the authorization to access the other service and the actions of other service can be controlled via AWS identity and access management **(IAM) role**. Each IAM role is attached with at least one **policy** that grants permissions to access resources specified in the action of the rule or to control actions toward the received data. For example, when an Amazon SNS rule is created, an IAM role will be attached to that SNS rule to authorize access to SNS resources. The attached role will have policies that allow actions (e.g. sns:Publish) toward specific resources in Amazon SNS. Similarly, when a lambda rule is created, an IAM role will be attached to the lambda function. This attached IAM role will have policies that authorize actions (e.g. iot:Publish, iot:GetThingShadow) toward specific resources in AWS Lambda. Thus, we can see that IAM role and its attached policies are a part of the AWS IoT rule definition to control actions. Figure 5.3 shows that each action of a rule

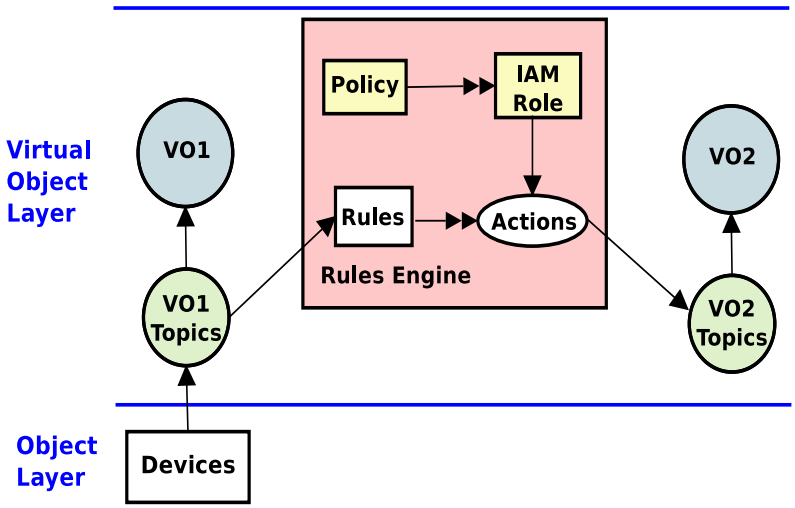

Figure 5.4: The Rules Engine as a Communication Channel in AWS-IoT-ACMVO

can only attach one IAM role, but each IAM role can be used by many rule actions. Also, one IAM role can attach many policies, and one policy can be attached to many IAM roles.

5.3 Issues in enforcing ACO-IoT-ACMsVO within AWS-IoT-ACMVO

AWS IoT does not support direct communication among VOs, because a VO is only allowed to communicate directly with its reserved topics. The AWS-IoT-ACMVO model is one way to effect VOs communication via rules within AWS IoT. AWS-IoT-ACMVO keeps the transient data within the virtual object layer without persistent storage, while only data about actual speeding cars is propagated to the higher layers. Thus, the privacy of data can be preserved. All components of VOs communication that contribute in this communication are shown in Figure 5.5. Figure 5.4 shows how the rules engine of AWS IoT serves as a communication channel between VOs in the AWS-IoT-ACMVO model. The ACO-IoT-ACMsVO academic model assumes the communication regime shown in Figure 5.2 where the communication channel between two VOs is a shared topic to which VO1 publishes and VO2 subscribes. The rules engine enables a similar effect to be achieved in AWS IoT as shown in Figure 5.4.

The control points to authorize VOs communication via topics in ACO-IoT-ACMsVO is placed on both VO side and topic (T) side [6]. For example, in case of ACL-Cap operational model, a

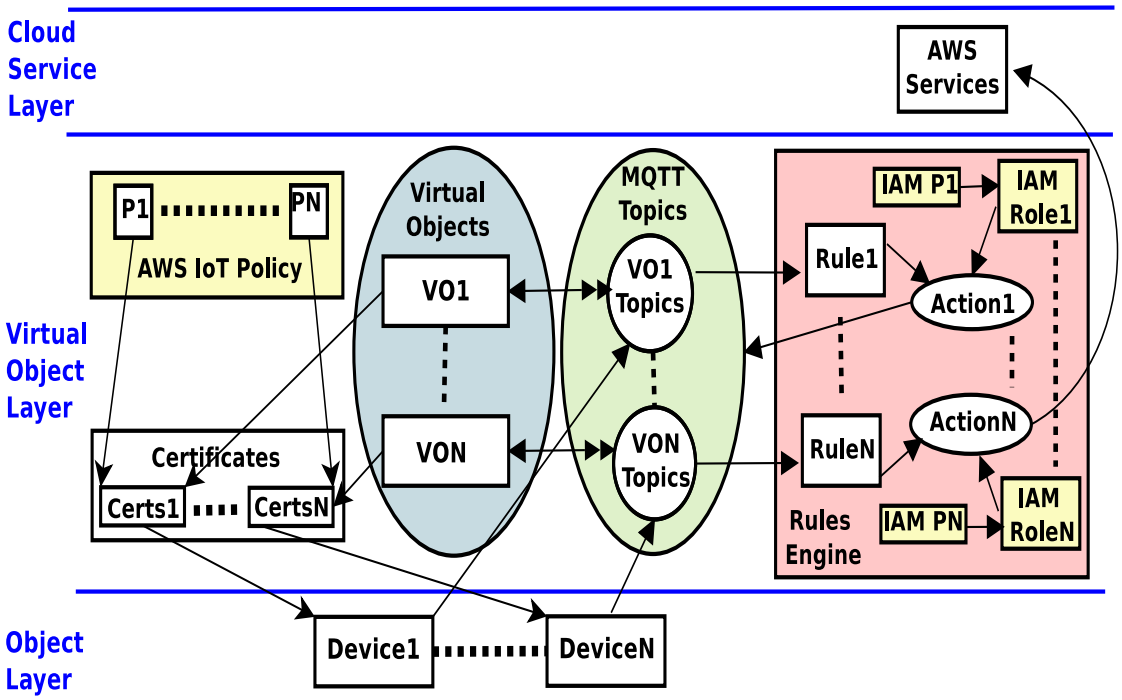

Figure 5.5: The Sensing Speeding Cars Use Case within AWS-IoT-ACMVO

VO can have a right to publish to a topic only if the topic in the capability list of the VO and the VO is in the access control list of T. In case of the ABAC operational model, a publish permission will be authorized if the topic is within VO-Publish attribute values and the VO is within T-Publish attribute values. The subscription right is similarly authorized on both sides.

In case of AWS-IoT-ACMVO, the control point to authorize reserved topics of a VO to communicate with other reserved topics of another VO is placed in rules engine. For example, when data arrives at reserved a topic of a VO, a lambda rule will trigger a lambda function as an action if the Select Clause and Where Clause in the SQL statement of the lambda rule evaluate to true. When the lambda function is triggered, an attached IAM role with lambda function will comply with a coupled policy or policies to authorize AWS-IoT to access to the lambda function and authorize the lambda function to execute actions with the received data. IAM role policy could authorize lambda function to forward data to other reserved/unreserved topics. Thus, other topics will receive data as long as it is in an appropriate format without checking where the data came from or rejecting the received data, and as a result, the received data will be forwarded to subscribers. So, a question like "which resources should a topic receives data from?" is only controlled via IAM role that is

attached within an action of a rule, and topics have no control over what they receive.

We investigated applying the use case of sensing speeding cars, which is employed in ACO-IoT-ACMsVO, within AWS-IoT-ACMVO. But as we discussed above, the communication style, access control points, access control models are not precisely alike. Although, AWSnIoT does not support direct VOs communication, we were able to develop AWS-IoT-ACMVO that effectively allows VOs communication. The details of configuration, scenario, and authorization policy are discussed in the following section.

5.4 A Use Case: Sensing Speeding Cars within AWS-IoT-ACMVO

In this section, we present two scenarios of the use case of sensing cars speed. The two scenarios will have number of sensors and a camera in the physical layer. All devices on the physical layer will push collected data to their virtual objects (shadows). In our scenarios, we focus on the communication among virtual objects and how this communication can be controlled.

5.4.1 Sensing the Speed of One Car

We will discuss the configuration and the scenario of our simple use case as follows.

Setup and Configuration

In this simple scenario, we will have two physical sensors and one physical camera each with one virtual object connected to it. Figure 5.6 shows the connected devices, virtual objects (shadows), certificates, AWS IoT policy, rules, actions and their IAM roles, and AWS services.

First, we create one virtual object for each physical object using AWS IoT management console and attach one X.509 certificate for each virtual object. For each certificate, we attached an AWS IoT policy. Certificates are copied into their corresponding physical objects to allow authentication and authorization of physical objects when they communicate with the corresponding virtual objects. In other words, the attached AWS IoT policy authorizes specific actions (connect and publish) for physical objects. When certificates are given to the corresponding physical objects,

they are accompanied by the private key of the certificate and an AWS root CA certificate.

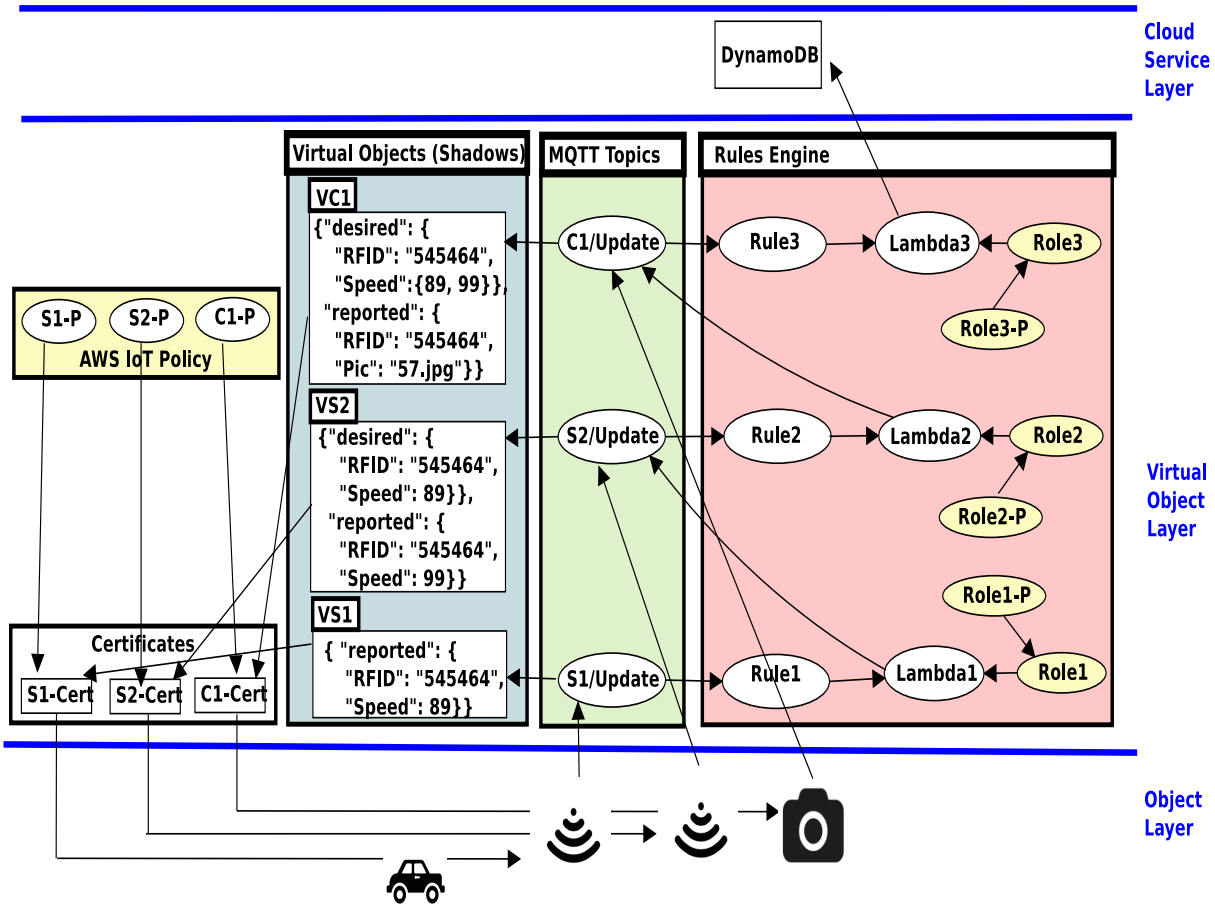

Figure 5.6: A Simple Use Case of Sensing the Speed of One Car

We simulated sensors and camera physical objects using AWS SDK for JavaScript (Node.js). There is an attached rule for each MQTT update topic $\$aws/things/Sensor_i/shadow/update$ that triggers a Lambda function. Lambda functions are responsible about republishing the incoming reported data that arrived to a virtual object ($Virtual\ Sensor_i$ or $Virtual\ Camera$) from its corresponding physical object ($Sensor_i$ or $Camera$) to the next virtual object ($Virtual\ Sensor_{(i+1)}$ or $Virtual\ Camera$) as shown in Figure 5.6. Also, each Lambda function is attached with IAM role that authorizes AWS IoT to access AWS and AWS IoT resources and services. The IAM role also controls Lambda function operations, such as republishing data to another topic or getting the current state of a shadow.

Senario

$Sensor_1$ sends RFID and Speed of the over speeding car as a $reported$ message to $Virtual\ Sensor_1$ ($VS1$) by publishing to $Sensor_1$ MQTT update topic $\$aws/things/Sensor_1/shadow/update$. $Rule_1$ that is attached with the $Sensor_1$ MQTT update topic will trigger $Lambda_1 function$ every time data arrived to MQTT update topic of $VS1$. $Lambda_1\ function$ republishes the arrived data to $Virtual\ Sensor_2$ ($VS2$) with a $desired$ tag. Figure 5.6 shows that the $reported$ RFID and Speed to $VS1$ is republished to $VS2$ as $desired$ state by $Lambda_1\ function$.

$Sensor_2$ also sends RFID and Speed of the over speeding car as a $reported$ message to $VS2$ by publishing to the following MQTT update topic: $\$aws/things/Sensor_2/shadow/update$. $Rule_2$ is going to trigger $Lambda_2\ function$ every time data arrived to MQTT update topic of $VS2$. $Lambda_2\ function$ check if the coming data is with $reported$ tag, it compares the saved $desired$ RFID with the coming $reported$ RFID from $Sensor_2$. If the two RFIDs are matched, $Lambda_2\ function$ combines the two speeds and one RFID and publish it with a $desired$ tag to the $Virtual\ Camera$ ($VC1$). Figure 5.6 shows that the $reported$ RFID matches the $desired$ RFID in $Virtual\ Sensor_2$. Thus, $VC1$ will receive from $Lambda_2\ function$ two speeds that are reported from $Sensor_1$ and $Sensor_2$ for the same RFID.

$Camera$ also sends RFIDs and pictures (Pic) of the passed cars as a $reported$ message to $Virtual\ Camera$ by publishing to MQTT update topic $\$aws/things/Camera/shadow/update$. $Rule_3$ is going to trigger $Lambda_3\ function$ every time data arrived to MQTT update topic of $VC1$. $Lambda_3\ function$ check if the coming data is with a $reported$ tag, it compares the saved coming $desired$ RFID from $Sensor_2$ with the coming $reported$ RFID from $Camera$. If the two RFIDs are matched, $Lambda_3\ function$ combines the RFID, Speeds, and Pic and store them to the Amazon DynamoDB. Figure 5.6 shows that the $reported$ RFID matches the $desired$ RFID in $VC1$. Thus, the combined RFID, Speeds, and Pic will be stored in in the Amazon DynamoDB.

```
{ "Version": "2012-10-17",
 "Statement":
 [
   { "Effect": "Allow",
     "Action": [ "iot:Connect" ],
     "Resource": ["arn:aws:iot:us-west-2:760000000000:
      client/Sensor2"]
   },
   { "Effect": "Allow",
     "Action": [ "iot:Publish"],
     "Resource": [ "arn:aws:iot:us-west-2:76000000000:
       topic/$aws/things/Sensor2/shadow/update"]
   }
 ]
}
```

Figure 5.7: $S2$-P that is Attached to $S2$-$Cert$

```
{
   "Version": "2012-10-17",
   "Statement": [
      {  "Effect": "Allow",
         "Action": "iot:GetThingShadow",
         "Resource": "arn:aws:iot:us-west-2:760000000000:
          thing/Sensor2"
      },
      {  "Effect": "Allow",
         "Action": "iot:Publish",
         "Resource": "arn:aws:iot:us-west-2:760000000000:
          topic/$aws/things/Camera/shadow/update"
      }
   ]
}
```

Figure 5.8: $Role_2$ $Policy$ that is Attached to $Role_2$

Authorization policy

There is an AWS IoT policy attached with each certificate to authorize specific actions for physical objects. For example, $Sensor_1$ are only allowed to connect and publish to $VS1$ in order to send the collected RFID and Speed of the over speed cars. Thus, the AWS IoT $S1$-P and $VS1$ are attached with $S1$-$Cert$ which is copied to $Sensor_1$. The policy states that connect and publish actions are allowed to the specified resources, which is $VS1$ (the shadow of $Sensor_1$). Similarly, the AWS IoT $S2$-P in Figure 5.7 and $VS2$ will be attached to $S2$-$Cert$, which is copied to $Sensor_2$, and the

AWS IoT $C1$-P and $VC1$ will be attached to $C1$-$Cert$, which is copied to $Camera_1$. AWS IoT defines policy variables, which can be used in AWS IoT policies within the resource or condition block. The basic variable $IoT : ClientID$ can be used to generate a policy that can be attached to all certificates. However, a certificate is not coupled with an ID of physical sensor that should connect and publish to the attached shadows, so malicious sensor could change their ID to connect and publish to any other MQTT update topic. Therefore, we preferred to specify and hard-coded one different policy for each certificate as shown in Figure 5.6, and each AWS IoT policy is similar to $S2$-P shown in Figure 5.7 but with different sensor names.

Also, there is an IAM role attached to each Lambda function to authorize it accessing to AWS services and AWS IoT resources. For example, $Role_1$ is attached to $Lambda_1$ to authorize it publishing to the update topic of $VS2$. Also, $Role_2$ is attached to $Lambda_2$ to authorize it getting the $desired$ state of $VS2$ and publishing to the MQTT update topic of $VC1$. Figure 5.8 shows the IAM $Role_2\ Policy$ that is attached to $Role_2$. Also, $Role_3$ is attached to $Lambda_3$ to authorize it getting the $desired$ state of $VC1$ and publishing to Amazon DynamoDB.

5.4.2 Sensing the Speed of Multiple Cars

The previous simple use case introduces the basic idea of implementing and controlling the virtual object communication within AWS IoT. However, in reality there is a need to track multiple cars, where different cars pass a sensor at a time. A VO (shadow) in AWS IoT has different reserved topics that are used by the VO to subscribe to them. So, any time a sensor publishes a new list of RFIDs/Speeds, the old list is deleted and a new one is saved. However, our use case with multiple cars needs to keep track of the historical data (old and new RFIDs).

In this use case, for every VO corresponding to a physical object, we propose to have another relative VO that works as storage of historical data. The only way to push or get data from the VO storage is by using a lambda function that is triggered by publishing data from a sensor to the MQTT update topic of the corresponding VO. Figure 5.9 shows sensors $(S1, S2, ..., Sn, C1)$ and their corresponding virtual objects $(VS1, VS2, ..., VSn, VC1)$ and the storage for each of them

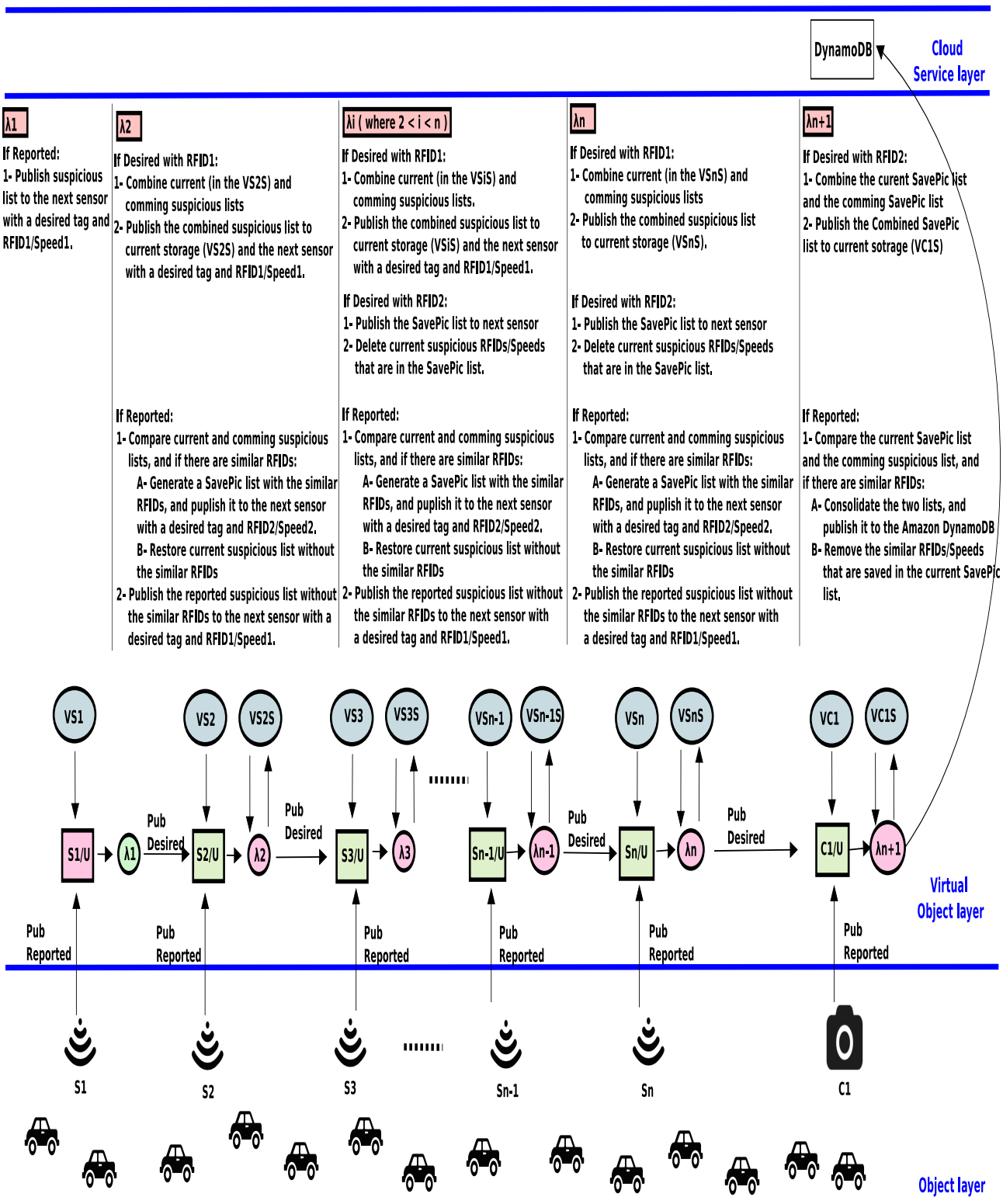

Figure 5.9: A Use Case of Sensing the Speed of Multiple Cars

$(VS1S, VS2S, ..., VSnS, VC1S)$.

Setup and Configuration

As in previous simple use case, we create one virtual object and one virtual object storage for our physical objects and attach one X.509 certificate for each virtual object. Certificates that are attached with AWS IoT policies are copied into their corresponding physical objects. The AWS IoT policy states that sensors and the Camera are only allowed to connect and publish to the corresponding VO (similar to the mentioned policy in Figure 5.7).

We simulated Sensors and the Camera using AWS SDK for JavaScript (Node.js). Lambda functions are triggered by rules that are attached with MQTT update topics of VOs. For example, $Lambda_1$ is triggered by rule1 that is attached to MQTT update topic of VS1. In general, Lambda functions are responsible about the complex computations, such as getting the stored data, comparing and consolidating the coming and the stored data, and republishing data to the current storage or next VO. Figure 5.9 describes the functionality of each lambda function.

Senario

$Sensor_1$ sends a list of RFIDs/Speeds of over speeding cars as a $reported$ message to $VS1$ by publishing to $VS1$ MQTT update topic. $Rule_1$ triggers $Lambda_1$ $function$ when a published request arrived to MQTT update topic. $Lambda_1$ $function$ will consider the reported RFIDs/Speeds as a suspicious list, that will be handled as described in Figure 5.9.

$Sensor_2$, ..., $Sensor_i$, .., $Sensor_n$ also send a list of RFIDs/Speeds of over speeding cars as a $reported$ message to their corresponding VO by publishing to VO_i MQTT update topic, where $2 \leq i \leq n$. The reported RFIDs/Speeds from physical objects is considered as a suspicious list (stored in $VSiS$ under reported tag) beside the suspicious list that is coming from a previous VO (stored in $VSiS$ under desired tag with RFID1). The matched RFIDs in both of the suspicious lists will be stored as SavePic list (stored in $VSiS$ under desired tag with RFID2). $Rule_i$ triggers $Lambda_i$ $function$ when a published request arrived to MQTT update topic of VS_i. $Lambda_i$

```
{"Version": "2012-10-17",
   "Statement": [
      {  "Effect": "Allow",
         "Action": "iot:Publish",
         "Resource": "arn:aws:iot:us-west-2:760000000000:
          topic/$aws/things/Sensor5_Storage/shadow/update"
      },
      {  "Effect": "Allow",
         "Action": "iot:GetThingShadow",
         "Resource": "arn:aws:iot:us-west-2:769000000000:
          thing/Sensor5_Storage"
      },
      {  "Effect": "Allow",
         "Action": "iot:Publish",
         "Resource": "arn:aws:iot:us-west-2:769000000000:
          topic/$aws/things/Camera/shadow/update"}
   ]
}
```


Figure 5.10: $Role_5$ $Policy$ that is attached to $Lambda_5$

$function$ will deal with the arrived data as suspicious lists and handle it to generate the SavePic list as described in Figure 5.9. Note that $Lambda_3$ to $Lambda_{(n-1)}$ will do the same computations.

$Camera$ sends RFIDs and pictures (Pic) of the passed cars as a $reported$ message to $Virtual$ $Camera$ ($VC1$) by publishing to MQTT update topic of $VC1$. $Rule_{(n+1)}$ triggers $Lambda_{(n+1)}$ $function$ when a published request arrived to $VC1$ MQTT update topic. $Lambda_{(n+1)}$ $function$ will deal with the coming data as descried in Figure 5.9.

Authorization policy

As in previous simple use case, we will have an AWS IoT $Policy$ that is attached with $S1$-$Cert$, .., Sn-$Cert$, $C1$-$Cert$. The policy state that physical objects can only connect to their corresponding VO and publish to MQTT update topic of the VO. Figure 5.7 is an example of an attached AWS IoT policy that authorizes physical objects to connect and publish.

Also, the IAM roles are attached to Lambda functions. For example, $Role_1$ is attached to $Lambda_1$ to authorize it publishing to the update topic of $VS2$. $Role_2$ is attached to $Lambda_2$ to authorize it getting the data of the storage of $VS2$ and publishing only to the update topic of the $VS2S$ and $VS3$. $Role_{(n+1)}$ is attached to $Lambda_{(n+1)}$ to authorize it getting the data of the storage of $VC1$ and publishing only to its storage and then to the Amazon DynamoDB. Figure 5.10

shows the $Role_5\ Policy$ of the IAM $Role_5$ that is attached to $Lambda_5$ to authorize it getting the data that is saved in the storage of $VS5$ ($n = 5$ in our implementation) and publishing only to the update topic of the $VS5S$ and to the update topic of $VC1$.

5.5 Performance

Our scenario propagates the Suspicious list published by any sensor until the last virtual sensor, and it propagates the SavePic list from the moment of generation until the camera. The first possible generated Suspicious list starts from $Sensor_1$, and the first possible SavePic list starts when $Sensor_2$ publishes similar Suspicious list to the published Suspicious list by $Sensor_1$, so the SavePic list will be generated by $lambda_2\ function$ that is triggered when $Sensor_2$ publishes to its virtual object. In this section, we calculate the time of propagating the Suspicious and the SavePic list to their final destination.

The use case with multiple sensors and cars is employed in computing the propagation time. we set the number of sensors to five. We used two AWS SDKs for JavaScript (Node.js) to subscribe to $Virtual\ Sensor_5\ Storage$ ($VS5S$) and $Virtual\ Camera_1\ Storage$ ($VC1S$), so we can get an acknowledgement whenever the Suspicious and the SavePic list are reached. A bash script is written to run $Sensor_1$, start the timer, run $VS5S$, and end the timer whenever we get an acknowledgement from $VS5S$. Similarly, the bash script will run $Sensor_2$ (with smillar RFIDs of $Sensor_1$), start the timer, run $VC1S$, and end the timer whenever we get an acknowledgement from $VC1S$. Thus, we were able to calculate the propagation time of the Suspicious and the SavePic list to their final destination.

We run $Sensor_1$ that publish the Suspicious list with $\{1, 10, 20, 30, 40\}$ RFIDS. For the Suspicious list with one RFID, we calculate the propagation average time of 10 times run. Thus, the propagation time of the Suspicious list with one RFID from S_1 until $VS5S$ in Figure 5.11, which is 5915 millisecond, is the average of 10 times run. Similarly, the propagation time of the Suspicious list with 10, 20, 30, 40 RFIDs from S_1 until $VS5S$, which is 6335, 7131, 7519, and 8109 millisecond, is also the average of 10 times run. However, we get rid of outliers, which is the time

values that exceed 10000 or less than 3000 millisecond.

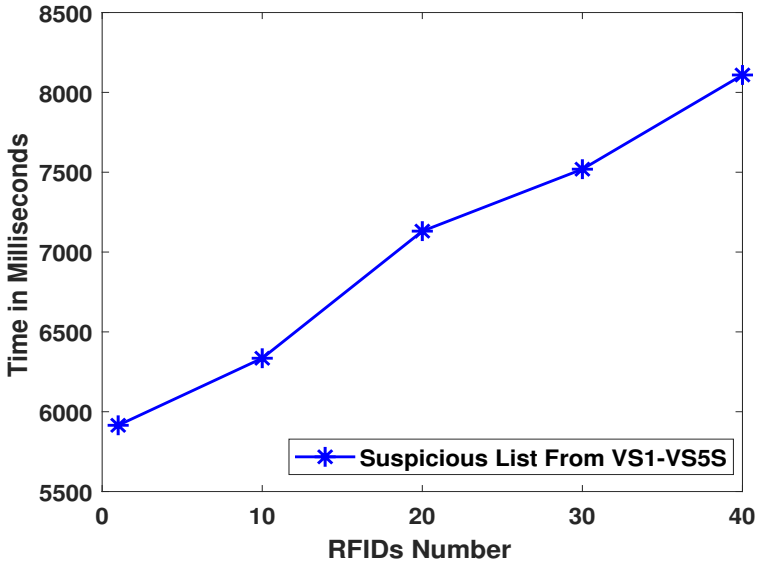

Figure 5.11: Propagation Time of Suspicious List from S_1 until $VS5S$

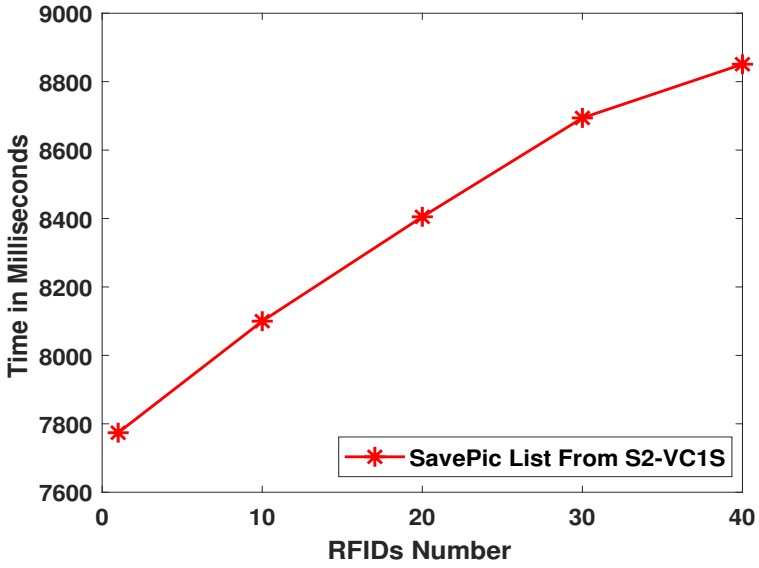

Figure 5.12: Propagation Time of SavePic List from S_2 until $VC1S$

After a Suspicious list is published by S_1 and an acknowledgement is received from $VS5S$, $Sensor_2$ is also run to publish a Suspicious list, similar to the Suspicious list that is published by S_1, with $\{1, 10, 20, 30, 40\}$ RFIDS. The propagation time of the SavePic list with $\{1, 10, 20, 30, 40\}$ RFID from S_2 until $VC1S$ in Figure 5.12, which is 7774, 8100 , 8405, 8694 , 8851 millisecond, is

the average of 10 times run. However, we get rid off outliers, which is the time values that exceed 14000 or less than 4000 millisecond.

The algorithms of our Lambda functions that we used within our use case in Figure 5.9 shows more computation and steps when a Lambda function gets the Suspicious list than when a Lambda function gets the SavePic list. However, our results in Figure 5.11 and 5.12 show that the propagation time of the Suspicious lists are less than the propagation time of the SavePic lists. This difference is because of the larger payload of the SavePic list, which has two speeds for each one RFID, than the Suspicious list, which has only one speed for each RFID.

5.6 Discussion

AWS IoT does not have full capability to implement our use case that we employed in [6]. First, virtual objects (shadows) in AWS IoT cannot communicate directly to each other. Since virtual objects can only subscribe to their reserved topics update, get, and delete. So, they receive data through their reserved topics. Also, virtual objects can publish to their reserved topics only whenever they receive data. As a result, publishing and subscribing of virtual objects is only to their reserved topics and direct publishing to unreserved topics or irrelevant topics is not applicable.

There are several indirect ways to allow virtual objects communication in AWS IoT. One way to allow two virtual objects to communicate is to attach a rule with the update topic of first virtual object that triggers a republish action to the second virtual object update topic. The Republish action can be also used to forward data to AWS services as shown in Figure 5.13. Another way is to attach a rule with a topic of first virtual object that trigger a lambda function, which can do complex computation, such as publishing data, getting data, and comparing data. Thus, lambda function can republish the received data to another topic, which could be the update topic of the second virtual object. We employed the second way in our use cases.

In addition to indirect communication among virtual objects, virtual objects in AWS IoT cannot keep track of old data. For example, if a new suspicious list is published to a virtual object, the current suspicious list will be deleted and the new one will be saved. However, our use case needs

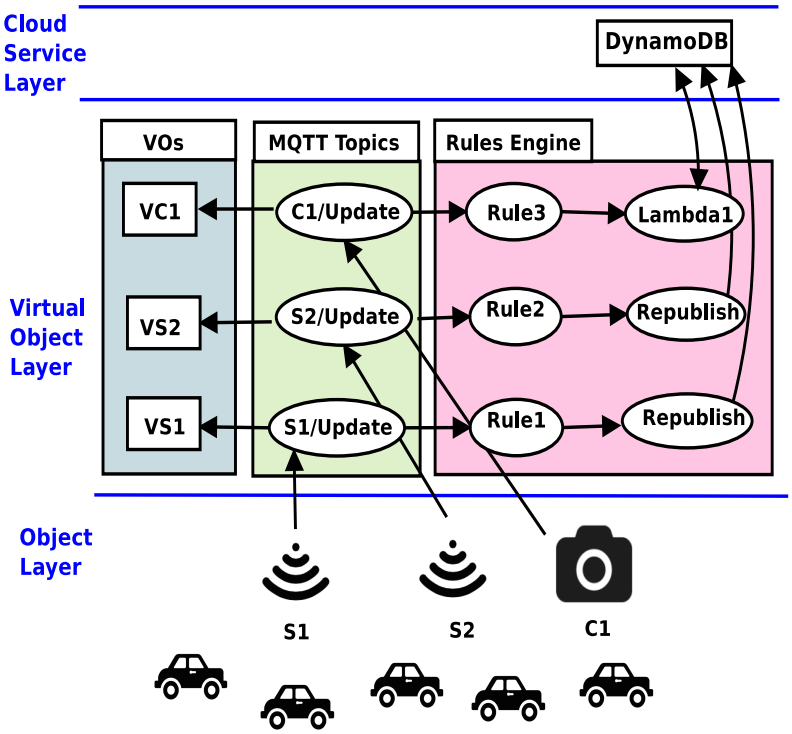

Figure 5.13: A Different Way of VOs Communication and Data Computation

to combine the coming suspicious list from previous sensor and the current saved one. Since the process of deleting and saving list is very fast, triggering a lambda function that get the current suspicious list from virtual object and then combine it with the coming one did not work. Thus, virtual objects in AWS IoT cannot save old data.

There are several ways to keep track of historical data in AWS IoT. One way to keep track of historical data of a virtual object is to have another relative virtual object that works as storage. The only way to get or publish data to the relative virtual object is by allowing one lambda function to publish and get data from it. This lambda function is triggered whenever data is published to update topic of the virtual object. Thus, the *coming* suspicious list arrived to the update topic of a virtual and the *current* suspicious list that is saved in the virtual object can be combined and republished by the lambda function. We used this way in our use case to keep transit data within the virtual object layer, so the privacy of data can be preserved. Another way to reach the historical data of a virtual object is to trigger a republish action to AWS DynamoDB whenever data is published to update topic of the virtual object. Thus, authorized virtual objects can get the historical data

from AWS DynamoDB as needed. However, our use case tends to keep the suspicious lists within the virtual object until at least two sensors report the speed of a car to be over limit. Figure 5.13 shows the way of republishing suspicious lists, which come from S_1 and S_2, to AWS DynamoDB in the cloud service layer. Then, $lambda_1$ is authorized to get all suspicious lists and check if there are duplicated RFIDs within the saved suspicious lists and also within the suspicious list coming from the camera. If so, this RFID is declared to be an over-limit car, and it is reported along with consolidate information from all suspicious lists (speed, picture).

Another issue with the AWS IoT is that virtual objects cannot do complex computation on the data they receive. They only save the recent published desired or reported data. Such a computation in our use case cannot be implemented within only AWS IoT. Thus, since we need the transit data to be only within AWS IoT, we used AWS Lambda service to support doing the needed computation. Another way to do that is to send data to DynamoDB and allow an application or a third party to do the needed computation.

Moreover, in our use case, we could have up to n sensors. $Lambda_3$ to $Lambda_{(n-1)}$ functions are repetitive functions that can be triggered by the update topic of $VS3$ to $VS_{(n-1)}$. We can get rid of this repetition If we have only one Lambda that accept passing the name of the published sensor. However, triggering same lambda is not working within our use case, because to our knowledge there is no way to pass the name of the published sensor to a lambda function. Thus, repeated copies of $Lambda_{(n-1)}$ will be increased by increasing the number of sensors, which is n.

Chapter 6: CONCLUSION

The following sections summarize the contribution of this dissertation and discuss some future research directions that can be further studied.

6.1 Summary

In this dissertation, we take a first step toward our eventual goal of evolving an authoritative family of access control models for cloud-enabled Internet of things. First, we developed an IoT architecture which is divided into four layers: the object layer, the virtual object layer, the cloud layer, and the application layer. This architecture will be our reference to build access control models for cloud-enabled Internet of things. We discussed illustrative examples that highlight the needed access control models for IoT. From our examples, we discussed the research agenda that could be studied.

We also used our ACO architecture to propose the ACL-Cap and ABAC operational models to control virtual object communication. We noted the unsuitability of conventional RBAC for this purpose. We illustrated the operational models by means of a use case involving sensors, camera and speeding cars. We further developed ACL, RBAC, and ABAC administrative access control models in context of this use case, and identified the advantages offered by progressing to more sophisticated models in this regard. Finally, assessment of security and privacy objectives for IoT, as identifed by Ouaddah et al [54], are discussed in context of our ACO architecture, access control models, and our use case.

Finally, we studied AWS IoT and developed the access control model for virtual objects (shadows) communication in AWS IoT (AWS-IoT-ACMVO). We used the AWS-IoT-ACMVO to implement two scenarios of the use case that is employed in ACO-IoT-ACMsVO: the simple use case of sensing the speed of one car with two sensors and the use case of sensing the speed of multiple cars with multiple sensors. By implementing these two scenarios using ACO-IoT-ACMsVO, we determined how to configure the policies and control virtual object communication of our proposed

model. The time to propagate information about suspicious cars and over-limit cars through all virtual objects is measured and discussed. Finally, upon our study and implementation, we offered a discussion of AWS IoT issues and suggestions of enhancing VOs communication and their access control.

6.2 Future Work

Our use cases within ACO reveals different possibilities of communications among entities in each layer and in different layers. As a result of these communications, the collected data flows among entities in various layers. From our ACO architecture and user cases, we recognized two major issues that need to be controlled: communications among entities, and data that flows through these communications. Figure 3.5 in Chapter 3 represents the general two main recognized issues and entities in each of them.

6.2.1 Controlling Communications

Access control models have been frequently used to control data access. On the other hand, enforcing access controls to determine what kind of communication or traffic is allowed onto the network has been less frequently discussed by researchers although widely used in practice in firewalls [55]. A firewalls is a decision and enforcement point that grants or rejects any communication flow through it.

In general, communications between entities at different layers are possible in different directions. In our ACO architecture, objects can communicate directly with each other at object layer. Many protocols have been proposed for networked devices communications, such as Bluetooth and WiFi [27, 48]. In addition, objects can communicate with their virtual objects with different associations. These kinds of communications introduce questions such as, which objects are authorized to communicate with a specific object? Which objects are allowed to access specific virtual objects? What are the necessary requirements for objects to authorize them to communicate? Is the collected data from objects going to participate in controlling the communication of an object?

All these questions can be studied and solved by proposing access control models for all kinds of objects' communications.

Our ACO architecture integrates the cloud as a service management layer, which helps in solving issues with IoT technology. It exists in the middle between the top and the bottom of the ACO architecture, so it is a crucial communication point that top and bottom entities should both access to communicate with each other. How do virtual objects get permissions to access each other or to access the cloud? Are virtual objects permitted to communicate directly with cloud entities or not? And what are the conditions and requirements for this communication? Do clouds communicate to share their information with each other? Can virtual objects be controlled or accessed through different (remote) clouds? Such questions should be addressed and appropriate access control provided.

ACO architecture allows users and administrators to remotely connect with IoT entities across applications in the application layer that generally display analyzed collected data. Also, applications can be used to control objects by sending commands that go from applications to cloud and virtual objects layers to control objects. Based on our ACO architecture, such communications between clouds, between virtual objects and clouds, and between applications and clouds can occur directly; communications between applications and objects, for example, should be transmitted through clouds and virtual objects. Figure 3.5 shows general entities that can communicate directly or through other layers and need to be controlled.

Within the IoT, indirect communication could introduce vulnerabilities in term of using an authorized communication to get unauthorized communication. For example, an application is allowed to communicate with a virtual object that has many-to-one association (many objects to one virtual objects). In contrast, the application is not allowed to communicate with some of the associated objects. Thus, such this association could cause indirect authorized access. It is important to have access control models to be used to control entities communication.

6.2.2 Controlling Access to Data

The ACO architecture integrates heterogeneous objects that collect data from an environment. Data could be collected by an individual object, such as the multi-value switch from our simple use case or a wearable FitBit device for one person. There could also be sub-data related to entities of the IoT, such as information about objects, virtual objects, and application. All sub-data and individual collected data can be accumulated and shared with others. Since our ACO architecture integrates the cloud, accumulated data is saved in cloud. Figure 3.5 shows that data is a result of communication, and also communication could be established to retrieve data. Thus, the relation between data and communication is bidirectional. It also shows that individual collected data and by object(s) and the sub-data are a subset of all accumulated data. Differentiating between who is allow to access the individual collected, sub-data, or accumulated data is necessary.

Data security should be applied at every stage of the data lifecycle because it is vulnerable from the moment of transferring it from the owner's data storage until it is deleted from cloud storage. According to [16], the data lifecycle is divided into seven stages: data generation, transfer, use, share, storage, archival, and destruction.

In every stage of data lifecycle, the confidentiality and integrity of this data is important. There are many questions raised regarding data security and privacy. Can an object, a virtual object, or an application access data partially or entirely? If so, can they retrieve data directly or across other entities? Can accumulated data in one cloud accessed by remote clouds, objects, applications, etc.? What is accumulated data used for? These and similar questions present themselves when dealing these issues.

6.2.3 General Issues

The virtual object layer includes virtual objects for physical objects. However, it is important to address issues such as whether virtual objects exist first or physical objects? Should both virtual objects and physical objects exist together or one of them could exist first? Can a virtual object appear without being a counterpart for any physical objects? These questions need be to be con-

sidered and addressed by studying and controlling the mapping between virtual objects and their physical objects.

The owner or the administrator of entities creates an access control rule to govern the set of allowable capabilities. For example, all virtual objects of wearable FitBit devices can view the average of the collected data from all wearable FitBit devices. Since the number of devices and applications in the IoT organization is unlimited, it is important for owners or administrators to apply access control policy without prior knowledge of particular entities that might require access [34]. ABAC allow an owner to implement access control policy without changing policy when new entities join.

Administrators control entities through applications that exist in the application layer. Since administrator and users both access through applications, it is important to distinguish administrators from users. How administrators control communications between entities at same and different layers? What kinds of actions that are self-control? What kinds of actions need direct control from administrators? All these questions need to be considered.

BIBLIOGRAPHY

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Comm. Surveys & Tutorials*, 17(4):2347–2376, 2015.

[2] Mohammad A Al-Kahtani and Ravi Sandhu. Rule-based RBAC with negative authorization. In *the 20th Annual Computer Security Applications Conference(ACSAC 2004)*, pages 405–415. IEEE, 2004.

[3] Asma Alshehri, James Benson, Farhan Patwa, and Ravi Sandhu. Access control model for virtual objects (shadows) communication for AWS internet of things. In *Proceedings of the Eighth ACM on Conference on Data and Application Security and Privacy*. ACM, 2018.

[4] Asma Alshehri and Ravi Sandhu. Access control models for cloud-enabled internet of things: A proposed architecture and research agenda. In *The 2nd IEEE International Conference on Collaboration and Internet Computing (CIC)*, pages 530–538. IEEE, 2016.

[5] Asma Alshehri and Ravi Sandhu. On the relationship between finite domain ABAM and $\text{PreUCON}_{\text{A}}$. In *International Conference on Network and System Security*, pages 333–346, 2016.

[6] Asma Alshehri and Ravi Sandhu. Access control models for virtual object communication in cloud-enabled IoT. In *The 18th International Conference on Information Reuse and Integration (IRI)*. IEEE, 2017.

[7] Amazon Web Services. AWS IoT features. "https://aws.amazon.com/iot/how-it-works/". Accessed Oct 2016.

[8] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, Oct 2010.

[9] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The social internet of things (SIoT) –when social networks meet the internet of things: Concept, architecture and network characterization. *Computer Networks*, 56(16):3594–3608, 2012.

[10] Jean Bacon, David M Eyers, Jatinder Singh, and Peter R Pietzuch. Access control in publish/subscribe systems. In *the Second International Conference on Distributed Event-Based Systems*, pages 23–34. ACM, 2008.

[11] Nathalie Baracaldo, Balaji Palanisamy, and James Joshi. Geo-social-rbac: A location-based socially aware access control framework. In *International Conference on Network and System Security*, pages 501–509. Springer, 2014.

[12] Smriti Bhatt, Farhan Patwa, and Ravi Sandhu. Access control model for AWS internet of things. In *International Conference on Network and System Security*, pages 721–736. Springer, 2017.

[13] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. On the integration of cloud computing and internet of things. In *IEEE Int. Conf. on Future Internet of Things and Cloud (FiCloud)*, pages 23–30, 2014.

[14] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[15] Imane Bouij-Pasquier, Anas Abou El Kalam, Abdellah Ait Ouahman, and Mina De Montfort. A security framework for internet of things. In *International Conference on Cryptology and Network Security*, pages 19–31. Springer, 2015.

[16] Deyan Chen and Hong Zhao. Data security and privacy protection issues in cloud computing. *2012 IEEE International Conference on Computer Science and Electronics Engineering (ICCSEE)*, pages 647–651, 2012.

[17] Yuan Cheng, Jaehong Park, and Ravi Sandhu. Relationship-based access control for online social networks: Beyond user-to-user relationships. In *IEEE Int. Conf. on Privacy, Security, Risk and Trust (PASSAT)*, pages 646–655, 2012.

[18] Yuan Cheng, Jaehong Park, and Ravi Sandhu. Attribute-aware relationship-based access control for online social networks. In *DBSec 2014: Data and Applications Security and Privacy XXVIII*, pages 292–306. Springer, 2014.

[19] COMPOSE. Collaborative open market to place objects at your service. "http://www.cloudwatchhub.eu/serviceoffers/compose-collaborative-open-market-place-objects-your-service". Accessed Sep 2016.

[20] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Trans. on Indust. Informatics*, 10(4):2233–2243, 2014.

[21] Ciprian Dobre and Fatos Xhafa. Intelligent services for big data science. *Future Generation Computer Systems*, 37:267–281, 2014.

[22] Mari Carmen Domingo. An overview of the internet of things for people with disabilities. *J. NW. & Comp. Appl.*, 35(2):584–596, 2012.

[23] Patrick Th Eugster and et al. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[24] Kosmatos Evangelos A, Tselikas Nikolaos D, and Boucouvalas Anthony C. Integrating RFIDs and smart objects into a unified internet of things architecture. *Advances in Internet of Things*, 2011:5–12.

[25] Sergei Evdokimov, Benjamin Fabian, Steffen Kunz, and Nina Schoenemann. Comparison of discovery service architectures for the internet of things. In *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, pages 237–244, 2010.

[26] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[27] Erina Ferro and Francesco Potorti. Bluetooth and Wi-Fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications*, 12(1):12–26, 2005.

[28] Li Gong. A secure identity-based capability system. In *1989 IEEE Symposium on Security and Privacy*, pages 56–63. IEEE, 1989.

[29] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[30] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling*, 58(5):1189–1205, 2013.

[31] Michael A Harrison, Walter L Ruzzo, and Jeffrey D Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[32] José L Hernández-Ramos, Antonio J Jara, Leandro Marın, and Antonio F Skarmeta. Distributed capability-based access control for the internet of things. *Journal of Internet Services and Information Security (JISIS)*, 3(3/4):1–16, 2013.

[33] V.C. Hu and et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162), 2013.

[34] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo. Attribute-based access control. *IEEE Computer*, 48(2):85–88, 2015.

[35] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. MQTT-S − a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software & Middleware (COMSWARE 2008)*, pages 791–798. IEEE, 2008.

[36] Infineon. Lifecycle management security at every step of the device lifecycle. "http://www.infineon.com/cms/en/applications/smart-card-and-security/internet-of-things-security/life-cycle-management/". Accessed Aug 2016.

[37] IoT-A. Internet of things - architecture. "http://www.iot-a.eu/public". Accessed Aug 2016.

[38] IoT-iCore. IoT-iCore: Empowering IoT through cognitive technologies. "http://ai-group.ds.unipi.gr/ai-group/node/2207". Accessed Aug 2016.

[39] Xiaolin Jia, Quanyuan Feng, Taihua Fan, and Quanshui Lei. RFID technology and its applications in internet of things (IoT). In *2nd IEEE Int. Conf. on Consumer Elect., Comm. and Networks (CECNet)*, pages 1282–1285, 2012.

[40] Xin Jin, Ram Krishnan, and Ravi Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 41–55. Springer, 2012.

[41] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Future internet: the internet of things architecture, possible applications and key challenges. In *10th IEEE Int. Conf. on Frontiers of IT*, pages 257–260, 2012.

[42] Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, 14(1):44–51, 2010.

[43] D Richard Kuhn, Edward J Coyne, and Timothy R Weil. Adding attributes to role-based access control. *IEEE Computer*, 43(6):79–81, 2010.

[44] Marc Langheinrich, Friedemann Mattern, Kay Römer, and Harald Vogt. First steps towards an event-based infrastructure for smart things. In *Ubiquitous Computing Workshop (PACT)*, 2000.

[45] Chi Harold Liu, Bo Yang, and Tiancheng Liu. Efficient naming, addressing and profile services in Internet-of-Things sensory environments. *Ad Hoc Networks*, 18:85–101, 2014.

[46] Jing Liu, Yang Xiao, and CL Philip Chen. Authentication and access control in the internet of things. In *ICDCS Workshops*, pages 588–592, 2012.

[47] Parikshit N et al Mahalle. Identity authentication and capability based access control (iacac) for the internet of things. *Journal of Cyber Security and Mobility*, 1(4):309–348, 2013.

[48] Patricia McDermott-Wells. What is Bluetooth? *IEEE Potentials*, 23(5):33–35, 2004.

[49] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[50] Dang Nguyen, Jaehong Park, and Ravi Sandhu. A provenance-based access control model for dynamic separation of duties. In *2013 Eleventh Annual International Conference on Privacy, Security and Trust (PST)*, pages 247–256. IEEE, 2013.

[51] Huansheng Ning and Ziou Wang. Future internet of things architecture: like mankind neural system or social organization framework? *IEEE Communications Letters*, 15(4):461–463, 2011.

[52] Michele Nitti, Virginia Pilloni, Giuseppe Colistra, and Luigi Atzori. The virtual object as a major element of the internet of things: a survey. *IEEE Communications Surveys & Tutorials*, 18(2):1228–1240, 2015.

[53] OASIS. *MQTT Version 3.1.1*. "http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/csprd02/mqtt-v3.1.1-csprd02.html". Accessed Jan 2018.

[54] 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*, 112:237–262, 2017.

[55] Paloalto Networks. What is a firewall? "https://www.paloaltonetworks.com/documentation/glossary/what-is-a-firewall". Accessed Aug 2016.

[56] Jaehong Park and Ravi Sandhu. The UCON_{ABC} usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[57] Pritee Parwekar. From internet of things towards cloud of things. In *2nd IEEE Int. Conf. on Comp. and Comm. Tech.*, pages 329–333, 2011.

[58] Pawani Porambage, Mika Ylianttila, Corinna Schmitt, Pardeep Kumar, Andrei Gurtov, and Athanasios V Vasilakos. The quest for privacy in the internet of things. *IEEE Cloud Computing*, 3(2):36–45, 2016.

[59] PV Rajkumar and Ravi Sandhu. Safety decidability for pre-authorization usage control with finite attribute domains. *IEEE Transactions on Dependable and Secure Computing*, 13(5):582–590, 2016.

[60] BB Prahlada Rao, Paval Saluia, Neetu Sharma, Ankit Mittal, and Shivay Veer Sharma. Cloud computing for internet of things and sensing based applications. In *Sixth IEEE Int. Conference on Sensing Technology (ICST)*, pages 374–380, 2012.

[61] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.

[62] Kay Römer, Thomas Schoch, Friedemann Mattern, and Thomas Dübendorfer. Smart identification frameworks for ubiquitous computing applications. *Wireless Networks*, 10(6):689–700, 2004.

[63] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):105–135, 1999.

[64] Ravi Sandhu, Edward J Coyne, Hal Feinstein, and Charles Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[65] Ravi S Sandhu. The typed access matrix model. In *In 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 122–136. IEEE, 1992.

[66] Ravi S. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, 1993.

[67] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.

[68] Chayan Sarkar, Akshay Uttama Nambi, R Venkatesha Prasad, Abdur Rahim, Ricardo Neisse, and Gianmarco Baldini. DIAT: A scalable distributed architecture for IoT. *IEEE Internet of Things Journal*, 2(3):230–239, 2015.

[69] SENSEI. SENSEI - I integrating the physical with the digital world of the network of the future. "http://cordis.europa.eu/pub/fp7/ict/docs/future-networks/projects-sensei-ec-summary_en.pdf". Accessed Aug 2016.

[70] Amazon Web Services. How the AWS IoT platform works - AWS. https://aws.amazon.com/iot/how-it-works/. Accessed April 2017.

[71] Dhananjay Singh, Gaurav Tripathi, and Antonio J Jara. A survey of internet-of-things: Future vision, architecture, challenges and services. In *Internet of things (WF-IoT), 2014 IEEE world forum on*, pages 287–292. IEEE, 2014.

[72] Jeffrey Voas. Networks of 'things'. *NIST Special Publication*, 800(183):800–183, 2016.

[73] Evan Welbourne, Leilani Battle, Garret Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. Building the internet of things using RFID: the RFID ecosystem experience. *IEEE Internet Computing*, 13(3):48–55, 2009.

[74] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du. Research on the architecture of internet of things. In *3rd IEEE Int. Conf. on Advanced Computer Theory and Engg. (ICACTE)*, volume 5, pages 484–487, 2010.

[75] Peng Wu, Yong Cui, Jianping Wu, Jiangchuan Liu, and Chris Metz. Transition from IPv4 to IPv6: A state-of-the-art survey. *IEEE Communications Surveys & Tutorials*, 15(3):1407–1424, 2013.

[76] Zhihong Yang, Yingzhao Yue, Yu Yang, Yufeng Peng, Xiaobo Wang, and Wenji Liu. Study and application on the architecture and key technologies for IoT. In *IEEE Int. Conf. on Multimedia Technology (ICMT)*, pages 747–751, 2011.

[77] Ning Ye, Yan Zhu, Ru-Chuan Wang, and Qiao-min Lin. An efficient authentication and access control scheme for perception layer of internet of things. *Applied Math. & Info. Sciences*, 9(4):1617–1624, 2014.

[78] Jianming Yong, Elisa Bertino, and Mark Toleman Dave Roberts. Extended RBAC with role attributes. In *PACIS 2006 Proceedings*, page 8, 2006.

[79] Xinwen Zhang, Yingjiu Li, and Divya Nalla. An attribute-based access matrix model. In *The 2005 ACM Symposium On Applied Computing (SAC?05)*, pages 359–363. ACM, 2005.

[80] Yun Zhang, Farhan Patwa, and Ravi Sandhu. Community-based secure information and resource sharing in AWS public cloud. In *2015 IEEE International Conference on Collaboration and Internet Computing (CIC)*, pages 46–53. IEEE, 2015.

[81] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu, and Weijun Qin. IoT gateway: Bridging wireless sensor networks into internet of things. In *8th IEEE/IFIP Int. Conf. on Embedded and Ubiquitous Comp.*, pages 347–352, 2010.

[82] Michele Zorzi, Alexander Gluhak, Sebastian Lange, and Alessandro Bassi. From today's intranet of things to a future internet of things: a wireless-and mobility-related view. *IEEE Wireless Communications*, 17(6), 2010.

VITA

Asma Hassan Alshehri was born in Riyadh, Saudi Arabia, in 1986, to Hassan and Zeina Alshehri. In Augest 2008, she earned her Bachelor?s degree in computer science from Princess Nourah bint Abdulrahman University College of science in Riyadh, Saudi Arabia. After graduation, in Fall 2008, Asma worked as a teacher for high school students. After that, Asma worked as a teaching assistant in King Saud University. In May 2010, she received a scholarship from King Saud University to come to the U.S. and pursue her Master's and PhD degrees. After studying intensive English at The University of Texas at Austin for one year, she began her Master's degree in August 2011 in the College of Science at The University of Texas at San Antonio. Asma received her Master of Computer Science from The University of Texas at San Antonio in December 2013. Asma entered the doctoral program in the Department of Computer Science at the University of Texas at San Antonio in Spring 2014. She joined the Institute for Cyber Security at UTSA and started working with Dr. Ravi Sandhu since Spring 2015. Her research interests include security and privacy in cyber space. In particular, her focus is on developing access control models for Cloud-Enabled Internet of Things.